VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

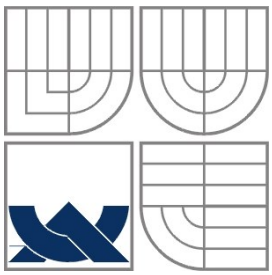
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILE SYSTEM FOR CUSTOMER FEEDBACK COLLECTION

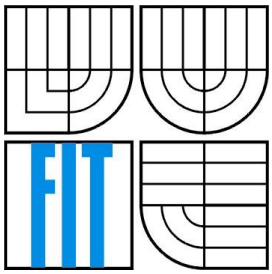
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JAKUB KADLUBIEC
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ SYSTÉM PRO SBĚR ZPĚTNÉ VAZBY ZÁKAZNÍKŮ

MOBILE SYSTEM FOR CUSTOMER FEEDBACK COLLECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB KADLUBIEC

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2013

Abstrakt

Práce se zabývá popisem tvorby mobilního systému pro monitoring zákaznické spokojenosti a sběr zpětné vazby od návštěvníků v restauracích s názvem Huerate. Komplexně jsou popsané všechny fáze vývoje systému. První část práce se zabývá analýzou existujících řešení a stavem na trhu. Následně jsou na základně komunikace s majiteli restaurací sestaveny požadavky na systém. Nakonec se práce věnuje samotnému návrhu systému, jeho implementaci a nasazení v restauracích. Systém Huerate běží jako webová aplikace a je dostupný na adrese http://huerate.cz.

Abstract

Thesis describes the process behind creating a mobile system for monitoring customer satisfaction and feedback collection in restaurants called Huerate. The creation of the system is described from the very beginning, where the existing solutions, situation on the market and user's requirements are analyzed based on communication with restaurant owners. After that, the design, implementation and deployment to the restaurants are described. System runs as a mobile web application and is available at http://huerate.cz.

Klíčová slova

mobilní webová aplikace, sběr zpětné vazby v restauracích, Scrum, cloud, Windows Azure, ASP.NET MVC, škálovatelnost, průzkum trhu, uživatelská přívětivost, Customer Experience Management, Entity Framework, dependency injection, SQL Azure, Azure Storage

Keywords

mobile web application, customer feedback collection in restaurants, Scrum, cloud, Windows Azure, ASP.NET MVC, scalability, market research, user friendliness, Customer Experience Management, Entity Framework, dependency injection, SQL Azure, Azure Storage

Citace

Kadlubiec Jakub: Mobile System for Customer Feedback Collection, diplomová práce, Brno, FIT VUT v Brně, 2013

Mobilní systém pro sběr zpětné vazby zákazníků

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Adama Herouta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Jakub Kadlubiec
7. ledna 2013

Poděkování

Děkuji vedoucímu panu Doc. Ing. Adamu Heroutovi, Ph.D. za poskytnuté cenné rady, ochotu a vstřícnost při řešení práce. Děkuji kolegům Romanovi Mazurovi a Romanovi Pittnerovi za spolupráci při tvorbě systému, zejména za pomoc s tvorbou uživatelského rozhrání.

Table of Contents

1 Introduction

According to the survey made among 315 restaurant owners in Czech Republic, 93% of them take guests opinions into account when managing their restaurants. On contrary, only 34% of them are satisfied with the amount and quality of feedback they get from their guests. This thesis aims to solve this problem by presenting an innovative tool for the feedback collection. The field of feedback collection is part of the broader discipline called Customer Experience Management. This discipline is used to ensure that customer is having the fluent experience while using some service [1].

This thesis describes the process behind initial research, requirements analysis, design, implementation and deployment of the mobile feedback collection system. The system is designed to be used in the restaurants, but in general can be used in any place where customers' opinion is needed to improve service. The core of the system is a web application optimized for mobile devices and running in the Windows Azure cloud and available publicly at http://huerate.cz.

Building an online feedback collection tool combines many challenging aspects at one place. At first, the user interface has to be designed in a very user-friendly way, so people with all levels of computer literacy would be able to use it intuitively. The other important thing is security, as stored and processed information are very sensitive. At last but not least, the system must implement correct statistical methods to display relevant results.

The challenge behind creating this system lays in fact that it will be used in real situations by people with different backgrounds and experience, and therefore the aspects mentioned in previous paragraph have to be taken very seriously and not only theoretically.

The name of the system is Huerate. It is a connection of words "hue" and "rate". "Rate" is used because system will be used to rate different aspects of the restaurants, while "hue" represents the color scale used to express the level of satisfaction.

At first, existing methods and tools for measuring customer satisfaction are analyzed. Results of the initial analysis are summarized in chapter 2. A survey among Czech restaurant owners was made to gather information about their methods for getting feedback from guests and to recognize their needs and requirements. After analyzing exiting methods for getting feedback from restaurant guests, the requirements for the mobile feedback collection system were analyzed. The result of this analysis is a prioritized list of requirements in chapter 2.2. This chapter also describes the Scrum methodology which was used to develop the whole system.

Chapter 4 describes design of the Huerate system from the technical standpoint. Focus is put mainly on the system architecture and user interface design. Also, the specifics of running a web application in the cloud and choices which have to be made when running applications there are written down in this chapter.

Chapter 4.7.3 describes how different parts of the system are implemented. Responsibilities and functions of each layer of the system are described. Interesting technical problems and their solutions are described in separate sections.

System was deployed to the production environment as soon as first usable version was ready. Fast deployment of the not-yet-perfect product allowed to test it in real life scenarios and to get feedback from restaurant owners as soon as possible. Using this approach, it was possible to alter system design based on the continuous feedback from early adopters. Chapter 6 describes the deployment process and evaluation of the testing in the restaurants.

Finally, the work done on this thesis is summarized. The whole thesis is concluded and analysis of parts which did and did not work out is made. The last chapter also explores possible future paths of improving Huerate system.

2 Existing Methods for Measuring Guest Satisfaction

According to survey made among 313 restaurant owners in Czech Republic, every one of them cares about opinions of their guests and tries to acquire them. However, the importance of the guests' opinions and the way of collecting them vary. The most popular way of collecting feedback from guests is through waiter. The waiter asks guests about their opinion and then passes feedback to the restaurant owner. According to the survey, 85% of Czech restaurants use this way of feedback collection. The second most popular way of getting feedback from guests is direct communication between restaurant owner and guests which is utilized in 73% restaurants. Table 1 displays methods of getting feedback from restaurant guests which are utilized in 313 Czech restaurants.

Method of feedback collection	No. of restaurants	% of restaurants
Waiter communicates with guests	257	82%
Restaurant owner communicates with guests	226	72%
Guests send e-mails with opinions	198	63%
Guests write opinions on Facebook	163	52%
Feedback form on restaurant's website	129	41%
Physical book of wishes and complaints	41	13%
Printed questionnaires in restaurants	41	13%
Link to electronic questionnaire in restaurants	4	2%
Other	36	12%

Table 1: Current methods of getting feedback from restaurant guests

Hidden behind Other is mainly reading reviews on websites like tripadvisor.com, foursquare.com, alacarte.cz, slevomat.cz or damejidlo.cz.

An interesting fact is that only 2% of Czech restaurants use some kind of electronic feedback solution advertised directly in the restaurants and thus aiming on users with smartphones. Low penetration of system similar to Huerate has its advantages and disadvantages. The obvious advantage is that there is a place for a new system in the market. On the other hand, restaurants and guests are not used to such system and teaching them to use it might be difficult.

Restaurant owners were asked about the number of opinions they receive from their guests in a month. The result numbers show that 80% of restaurant owners receive less than 20 opinions in a month and 30% receive only less than 5 opinions. Only 34% of restaurant owners are satisfied with number and quality of customer opinions they get. 45% of restaurant owners would like to get more opinions from their guests. These numbers show that online system for feedback collection can be a big help here. Another thing survey revealed is that restaurant owners value feedback from their guests very much. 93% of restaurant owners take guests opinions into account when managing their restaurants.

In addition to getting feedback directly from guests, 25% of restaurants use mystery shopping to recognize flaws in their service. Mystery shopping is described in detail in the next section.

Full results of the survey are shown in Appendix A.

2.1 Existing Solutions

There are number of online feedback collection solutions which are more or less applicable in the restaurants. These tools will be examined before coming up with requirements for the Huerate system in the next chapter. Offline solutions such as printed questionnaires will not be considered here.

Collecting customer's feedback is part of broader field called Customer Experience Management. This field focuses on the opinions but also the needs of the customer [1]. Huerate system will take care only of the feedback collection and the rest is up to restaurant's owner.

Each existing solution's functions, its business model and market penetration will be described. Information about business model and market penetration are sometimes impossible to get. In these cases, only the estimated numbers will be put down.

2.1.1 Multi-purpose, Web Based Survey Solutions

Web based survey solutions allow anybody to create their own survey and put it online. As these services are multi-purpose, they tend to be very customizable. In general, this level of customization is not needed in customer feedback collection but never the less, it can be used.

There is a great number of solutions which belong to this category but only the few most significant ones were selected to be examined.

SurveyMonkey

SurveyMonkey is the world's most popular online survey tool, partly because they are offering very broad feature set [2]. At the end of 2011, SurverMonkey acquired one of its biggest competitors called Zoomerang. This acquisition even strengthened its leading position [3].

The business model of SurveyMonkey is based on having free and premium account types. This model is called freemium and it allows users to use a service for free with limited features available while it earns money from paying users who require more features. The advantage of this model is that users who use free service are advertising the tool [4]. Usually using the free service has a requirement that aims at this goal. In SurveyMonkey's case it is the requirement that each free survey has to have text "Powered by SurveyMonkey" in its footer [5].

SurveyMonkey's free account is limited to creating surveys with the maximum of 10 questions and 100 responses. Their most popular plan called Gold has unlimited number of questions and responses per survey as well as some of the more advanced features like survey A/B testing and responses analysis. This plan costs 300EUR per year [5].

SurveyGizmo

SurveyGizmo's target audience are mainly marketers, consultants and business professionals. Its main advantage is that it is trying to be as easy to use as possible [6]. One of the most interesting features which differentiates SurveyGizmo from SurveyMonkey is that SurveyGizmo provides full API access to its surveys [7].

SurveyGizmo offers free account which is limited to 50 responses per month per survey. Their paid plan costs 810$ per year [7].

Google Forms

Google offers a tool for building surveys as part of its Drive apps. This tool is different from the previously listed in terms of business model because Google Forms is a free tool. Responses are saved to a Google spreadsheet and analysis is up to user [8].

Google Forms can be considered as lightweight, because it does not offer advanced features as SurveyMonkey or SurveyGizmo. However, it can still be very well used to collect customer feedback.

2.1.2 Customer Experience Management Solutions

Feedback collection solutions which are designed to be used in customer oriented environments (such as restaurants) will be examined in this section. They are part of the broader field called Customer Experience Management. It has to be said, that survey solutions described in previous section can be used in customer oriented environment as well, but those specially designed for this, have much more in common with Huerate system, so they deserve separate section. Tools in this section will be described in more details.

Doporucim.cz

Doporucim is a tool for active customer satisfaction and experience management. It is based in Czech Republic and operating mostly in Czech Republic, Slovakia and Poland.

Doporucim's main goal is getting continuous feedback from customers. The tool is designed to be used right when the customers are consuming services. Feedback can be collected via many channels, most importantly SMS messages and mobile optimized web form. The web form is accessible by scanning QR code.

Doporucim can be used in many segments, for example in retails, hotels, conferences and gastronomy. That is one of the differences between Huerate and Doporucim, because Huerate will be optimized only for restaurants. Another built-in feature of Doporucim is giving gifts after finishing the questionnaire.

Operator of the survey can view results in administrative interface. Results are presented in intuitive graphs which highlights possible issues. Alternatively, results can be viewed as individual records.

Service allows user to download ready to print leaflets with filled directions how to send feedback. Those leaflets are available in many themes and sizes, so they will be usable in the most situations.

Pricing of Doporucim is based on the freemium model. The free version allows only 50 answers monthly and survey can consist of 1 question only. There are two paid versions. One for 299kc monthly and more advanced one for 999kc monthly. Both of them allow unlimited number of answers. They mainly differ in number of so called codes, which can be used to create different surveys and in advanced features available [9].

Screenshot on Figure 1 shows how doporucim.cz looks on a smartphone browser.

Medallia

Medallia is a Silicon Valey based company which takes care of all aspects of the Customer Experience Management. Opposed to Doporucim, Medallia provides very complex solution and aims at large companies.

Medallia's system handles more than 700,000 logins each month. Its analytical tools use newest technologies based on NoSQL database and OLAP data warehouses to analyze such large amount of feedback [10].

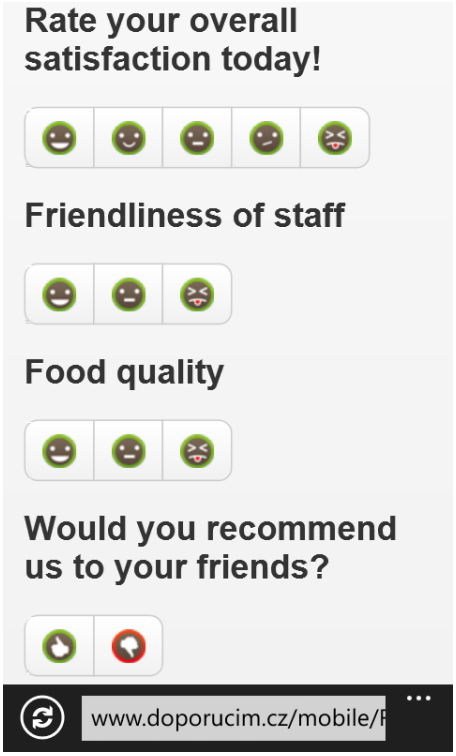


Figure 1: Doporucim.cz on a smartphone browser

2.1.3 Existing Solutions Overview

There was only a fraction of existing solutions presented in this chapter but the significant representatives were chosen, therefore it should give good overview of the market. It can be seen that there are a lot of different approaches of online feedback collection tools to choose from. The most basic tools are web surveys, which are very affordable, but leave the most of the work to the user. On the other hand, there are tools specializing at feedback collection, such as Doporucim or Medallia. Those tools are significantly more expensive, but can provide much more accurate results and better experience for customers.

A lot of interesting facts about typical business models and common features were discovered by studying existing solutions. Those findings will be extensively used when analyzing requirements for the Huerate system in the next chapter and designing it in chapter 4.

2.2 Mystery Shopping

Mystery shopping is a technique used to measure quality of service. It is described in a separate section, because feedback does not come from real guests. Instead, it works by sending trained professional into a restaurant where he pretends to be a common guest. This mystery shopper behaves naturally, orders food, talks with service, etc. During the whole visit, mystery shopper takes notes about things that he agreed upon with the restaurant owner and at the end he prepares detailed report about his experiences. Mystery shopping is very often provided by specialized agencies.

Mystery shopper may write down information in these areas:

- if and how long after entering the restaurant, the guest is greeted
- whether or not the stuff is friendly and kind

- whether or not the employee suggested any add-on sales
- whether or not the employee invited guest to come back to the restaurant
- cleanliness of restaurant and restaurant facilities
- speed of service
- compliance with company standards relating to service and restaurant appearance
- whether or not the service is able to solve an unexpected situations such as guest returning food

Mystery shopping can provide valuable feedback to the restaurant owner; however it can measure only technical issues. Things like overall satisfaction with the visit or whether the guest will recommend restaurant to his friends can be discovered only by communicating with actual guests and therefore mystery shopping has to be complemented by getting feedback from them [11].

According to the interviews with the Czech restaurant owners and survey made among them, only 25% of them use mystery shopping, mainly because of the high financial costs. Even if they use it, the average interval between visits of mystery guests is three months.

3 Requirements Analysis and Development Methodology

This chapter describes the process of requirements elicitation and analysis which led to creating project backlog. Analyzing requirements is very important step in the product lifecycle, because if done wrong, the whole effort put into developing the system can be useless. The most important requirements are the ones which solve users' problems, because only then users will be willing to use the system or even pay for it.

Gathering requirements had proven to be challenging, because users often did not say clearly what they need. Sometimes they did not even realize what they need. Coming up with a good set of requirements was even more difficult, because the products similar to the developed one are used by only a handful of restaurants and therefore there is no clear inspiration.

Because there is no easy way to come up with the perfect list of requirements on the very beginning, an iterative development methodology, namely Scrum, was chosen to develop the Huerate system. Scrum methodology divides development process into sprints, each having four weeks at maximum [12]. After every sprint the developed product should be ready to be deployed and used. This strategy is very effective, because it allows developed system to be used by early adopters very soon and immediately get their opinions about it. After this, requirements can be altered, priorities changed and so on. In the Huerate's case, the MVP[1] was deployed after less than one month of development. Detailed process of requirements elicitation is described in section 3.2. Gathered requirements are listed right after that in section 3.3.

Scrum methodology dictates how requirements look like and how they are organized and thus it is described before the requirements elicitation itself. Section 3.1 is devoted to describing basics of the Scrum development methodology and the way it was applied to the Huerate project. Particular focus will be placed on the requirements collection and creation of the product backlog.

3.1 Scrum Methodology

Scrum is an agile software development framework, which can be used in projects of any size. It is agile which makes it very well suited for projects, where long term planning is not possible. It builds on top of iterative and incremental development models. Product is developed in a flexible way, without any significant long-term planning. New versions are released periodically and after each release, product is tested and plan for next few weeks is made. Future users of the product are included in the development process from the very beginning and their opinions are evaluated after every release [12].

Main principles of agile development methodologies as described by Manifesto for Agile Software Development are those:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation

[1] Minimum Viable Product – a product with the absolute minimum of features allowing it just to be deployed and demonstrate basic principles.

4. Responding to change over following a plan

Items on the left side are valued more than items on the right side. This does not mean that things on the right side are not valued, they are just less important. Following agile principles was very important during the Huerate project development phase, because requirements were updated based on the interactions with users. Also, since the time span assigned for the development was not infinite, following a wrong plan for too long could have been fatal.

Scrum is best used for teams of more than 4 people and since the Huerate system was developed only by one developer, many parts were omitted. Basic Scrum terminology will be described in next section. After that, the application of Scrum to the development of the Huerate project will be described.

3.1.1 Basic Terminology

Main Scrum related terms will be shortly described in this section. Only the ones that are applicable to the one-person Scrum development will be listed.

Sprint

Sprint is a unit of development in Scrum. Sprint lasts a predefined amount of time, which should be specified in advance and should not change. Typical duration of one sprint is 2 to 4 weeks. At the end of sprint, there should be a functional product ready. It is very important, that requirements finished in a sprint are really functional and out of bugs. Only then, the real progress can be tracked.

At the beginning of the sprint, the development team chooses which requirements will be implemented in the upcoming sprint. This decision is being made on the Sprint Planning Meeting. The requirements to be implemented in next sprint are moved from Product Backlog to the Sprint Backlog.

After more sprints have passed, it is possible to know how much work can be done in one sprint and thus estimate accurately when the project will be ready. To achieve good accuracy, it is necessary to mark requirements as finished when they really are fully developed and tested [12].

There is an illustration of the Scrum development process in Figure 2.

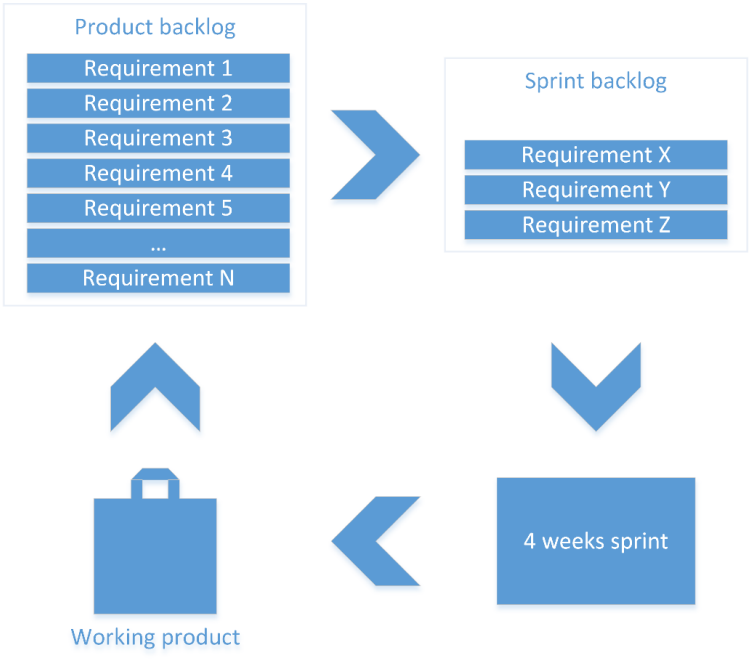


Figure 2: Illustration of the Scrum development process

Backlog

Backlog is a list of requirements which shall be implemented. Each requirement consists of the following fields:

- **Unique identifier (a name)**
- **User story**
  User story is a description of the requirement which puts value to the customer at first. It usually is in form "As *who*, I want to *what*, because *why*". Specifying requirements like this helps developers to realize, why they are implementing the feature.
- **Additional technical description**
  If it is not clear from the user story how to implement a requirement or what features are included, additional technical information are added to the requirement.
- **Priority**
  Priority is used to order requirements from the most important to the least. At sprint planning requirements with highest priority are usually taken first. Priority can be expressed by simply ordering the backlog from the most important issues to the least important ones.
- **Estimated complexity**
  This field is used to express how much effort will be needed to finish the requirement. Effort required is measured in story points. Story point is a relative unit. It does not relate directly to hours. It is only important, that requirement which is twice as complex as the other one has twice as many story points.

Scrum recognizes two types of backlogs:

- **Product backlog**
  All requirements which shall be implemented in the product.
- **Sprint backlog**
  Requirements which shall be implemented in the upcoming sprint. It is disallowed to add or remove items from this backlog during the sprint.

Burndown chart

Burndown chart is a tool used to visualize work progress and remaining time. There are sprints on the x axis and number of story points remaining on the y axis. Figure 3 shows how a sample burndown chart can look like.

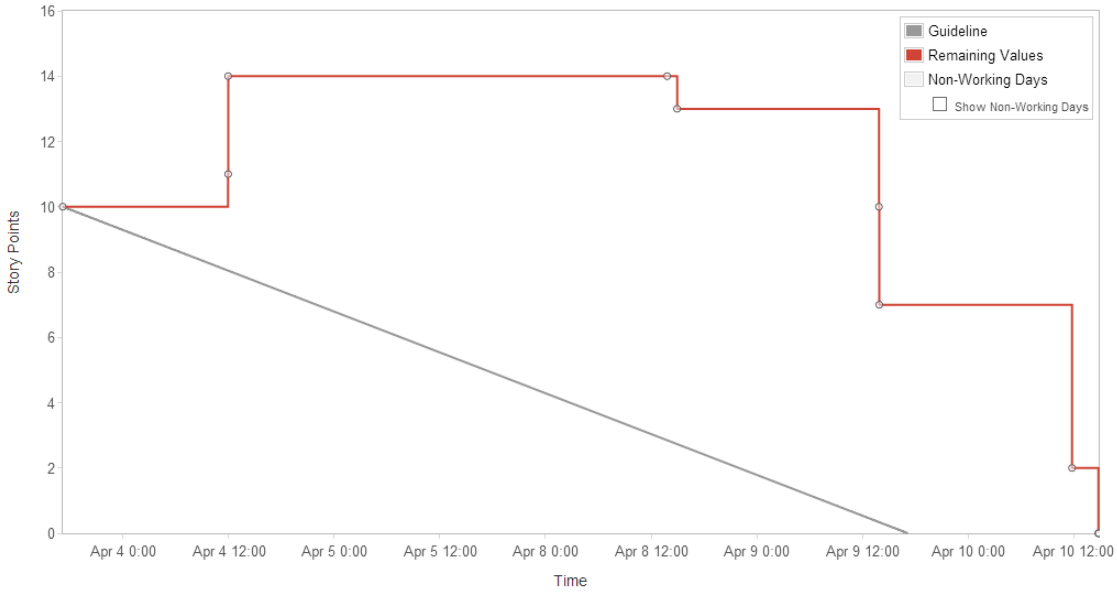


Figure 3: Burndown chart example as shown in Jira agile planning software

3.1.2 Application of Scrum on the Huerate Project

As Huerate project will be developed by only one developer, it is needed to alter the Scrum process to make it fit this kind of development. There are a lot of meetings in classic Scrum process mainly to synchronize team and improve processes. Those meetings are not necessary in one-person development and will not be held.

Sprint duration will be four weeks. The longest possible duration of the sprint was chosen, because there is only one developer and if sprint duration was shorter, no significant work could be finished in a sprint. Choosing four weeks sprints allows developing functional parts in one sprint.

The complexity of requirements is usually estimated by playing planning poker[2] with all team members. In Huerate case, estimates will be made by the only member of team which can lead to inaccuracies. To remove the complexity and minimize false sense of accuracy, the estimates will be chosen only between five levels:

- **Extra-small (XS)** – 1 story point
- **Small (S)** – 2 story points
- **Medium (M)** – 4 story points
- **Large (L)** – 13 story points
- **Extra-large (XL)** – 40 story points

The number of story points shows how complex the requirements are in relation to each other. For example requirement marked as Medium is four-times as complex as Extra-small requirement.

Extra-large requirements are usually too complex to be estimated accurately and should be split into smaller ones.

[2] Planning poker is estimating technique where all team members show at the same time a number of story points they think the feature deserves. If numbers do not match, short discussion takes place and after that another round of showing cards is played [13].

3.2 Requirements Elicitation

Requirements gathering is very important process, because when done wrong, the whole effort which comes next can be pointless. The goal of requirements elicitation is to identify requirements, which are the most important, e.g. which will make customers want to use (or even better to pay for) the software.

Requirements can be gathered from the various sources. The most obvious source are the future users. System has to solve users' problems. The issue here is that users sometimes do not realize what their problems are.

In Huerate case, the users are owners of the restaurants which will be the customers and will use the administration interface. The other kind of users are restaurant guests which will use the frontend of the application on their smartphones.

Interviews have been made with both groups of the users and most of the requirements came from those interviews. Owners of fifteen restaurants in Brno and three in Prague have been interviewed for 15 minutes on average. Also guests of those restaurants have been interviewed for shorter time each.

There was also one section devoted to the questions about features which are useful for the restaurant owners in the survey. The survey was made among 315 owners of the Czech restaurants and its full results are in Appendix A. Answers from the survey were also used extensively to assemble the final list of requirements. Another source of requirements were existing products as described in section 2.1.2.

3.3 Project Backlog of the Huerate System

Requirements for the Huerate system are listed in this section. Each requirement is structured as is described in section 3.1.1. Requirements were put together by methods described in section 3.2.

The development of the Huerate system was based on the Scrum methodology which means the progress was iterative and incremental. It applies to the design and development of the system as well as to the forming of the backlog. Majority of the requirements were put together at the very beginning; however some of them were added later. Priorities of the requirements were also changing as the project evolved. The list in this section is a snapshots of the backlog as it looked like at the end of the development phase. Displaying how the list changed after every sprint would not bring any interesting information, so it is not included in the thesis.

Requirements are grouped by the role which will benefit from implementing the requirement. Requirements are ordered by their priority. The ones listed at the top were implemented first.

3.3.1 Restaurant Owner

The first group of requirements consists of those which will bring benefit the owners of the restaurants who want to get feedback from their guests.

Administration interface

As a restaurant owner I want to be able to log into administration interface, because I want to manage my restaurant whenever I want.

This requirement consists of implementing a skeleton of an administration interface. Restaurant owner will be able to sign in to his administration by his email and password created when setting up an account. The interface has to be clean and intuitive, without unnecessary clutter. There will be a button to log out.

Complexity: Large
Priority: High

Viewing daily results
As a restaurant owner I want to view how guests rated my restaurant by days, because I want to know where I should improve and to compare results by days in order to examine differences between my stuff.

There will be a section in the administration which will show all questions and the average answer expressed in percent. Also the number of guests who answer that day will be shown. Text answers will be listed one by one.
Complexity: Medium
Priority: High

Getting feedback form URL
As a restaurant owner I want to give my guests the URL with the feedback form of my restaurant, because I am interested in their opinion about my service and thus I need to point them to my form.

Complexity: XS
Priority: High

Creating default feedback form after creating restaurant account
As a restaurant owner I want to have everything set up after I sign up my restaurant, because I do not want to waste time configuring form.

Survey among restaurant owners showed that they want to spend only minimal amount of time configuring their restaurant's form. Therefore, the default form with questions which are most needed according to the survey will be created for every newly created restaurant.
Complexity: S
Priority: High

Multilingual feedback form
As a restaurant owner I want to be able to define questions in more than one language, because my guests speak different languages.

This requirement is considered as very important, because a lot of restaurant owners said that they serve many customers from abroad and they cannot show them form only in Czech.
Complexity: M
Priority: High

Editing feedback form
As a restaurant owner I want to be able to edit questions in my form, because I want to ask my guests about things that really matters to me.

This feature will add a new section to the administration where restaurant owners will be able to completely adjust form according to their needs. Form consists of steps which can be reordered and removed. New steps can be added as well. There will be those types of steps available:

- Language selection
- Questions with sliders
- Yes/No questions

- Text questions
- Static text information
- Form ending

Texts in each of these steps can be configured. In steps containing questions (with sliders or Yes/No) it will be possible to edit those questions.

Restaurant owner will also be able to edit in which languages the form will be available and to edit all texts in each language separately.
Complexity: L
Priority: Medium

Secured form results
As a restaurant owner I need answers provided by my guests to remain private, because leaking them to the public can harm my business.

Complexity: M
Priority: Medium

Changing form URL and restaurant name
As a restaurant owner I need to be able to change link to my form, because I want to make it as short as possible.

Complexity: XS
Priority: Medium

Deleting account
As a restaurant owner I need to delete my account, if I no longer want to use the service, because I want to erase my data.

Complexity: S
Priority: Medium

Viewing total results
As a restaurant owner I want to view overall rating of my restaurant, because sometimes I do not want to browse results day by day.

Complexity: S
Priority: Medium

Getting results by email
As a restaurant owner I want to get periodically answers from my guests on email, because I do not want to waste time by visiting the administration all the time.

37% of restaurant owners said that their preferred way of viewing guest answers is by email.
Complexity: M
Priority: Medium

Being able to ask guests for their email address
As a restaurant owner I want to get email address of my guests, because I want to contact them later.

Implementing this requirement means to create new type of form step which will gather email address and another section in the administration interface where all email addresses would be listed. 72% of restaurant owners said that it would be useful to get guests' email addresses.
Complexity: S
Priority: Medium

Being able to ask opinions on an individual meals
As a restaurant owner I want to ask how individual meals tasted, because I want to know which meals I should improve.

76% of restaurant owners said they would like to know opinions on individual meals.
Complexity: M
Priority: Medium

Being able to ask opinions on an individual meals
As a restaurant owner I want to ask how individual meals tasted, because I want to know which meals I should improve.

76% of restaurant owners said they would like to know opinions on individual meals.
Complexity: M
Priority: Medium

Downloading QR code with link to form
As a restaurant owner I need to be able to download a QR code with line to my form encoded, because it is easier for guests to scan a code than to write address.

Complexity: S
Priority: Medium

Help page
As a restaurant owner I want to have a manual about the service usage, because I want to quickly solve my problems.

Complexity: S
Priority: Low

Recovering forgotten password
As a restaurant owner I need to have an option to recover forgotten password, because I want to sign into administration even if I have forgotten a password.

Complexity: S
Priority: Low

Being able to give guests a gift for filling a form
As a restaurant owner I want to offer my guest a gift for filling a form, because I want to get more answers and to please the customers.

Implementing this requirement means to create new type of form step which will display code of the gift and another section in the administration interface where all gifts would be listed. 46% of restaurant owners answered that they would like to have this feature.
Complexity: M
Priority: Low

Changing account information
As a restaurant owner I need to be able to change email address and password I use to sign in to the administration section, because I may have entered the wrong data when setting up my account.

Complexity: S
Priority: Low

3.3.2 Restaurant Guest

This group contains requirements which are needed in order to give restaurant guest an opportunity to express his opinion about the restaurant he is visiting at the moment. All features listed in this group have to be implemented with best user experience in mind, because restaurant guests might be people with all levels of computer literacy. The user interface has to be accessible on mobile browsers.

Giving feedback on a mobile device
As a restaurant guest I want to express my opinions about the restaurant, because I want it to become better or I want to receive a gift for filling the form.

This requirement covers creating a website accessible on mobile devices where guest will be able to say what he thinks about the restaurant, he is visiting right now. Every registered restaurant will own a unique link which will lead guest to the restaurant's form. Form will display questions specified by restaurant owner in his administration.

Size of the website has to be as small as possible, because it will possibly be accessed on a small mobile data networks. User interface has to be intuitive and fast, preferably without any page reloads.
Complexity: L
Priority: High

Selecting form's language
As a restaurant guest I want to select the language of the questions, because I might not speak Czech but I still want to give my feedback.

If restaurant owner makes his form available in more than one language, it has to be possible to choose the language of the form by the user.
Complexity: M
Priority: High

3.3.3 Huerate Owner

Requirements listed in this group are needed by the owner of the Huerate service in order to ensure fluent operation.

New restaurant registration
As a Huerate owner I want restaurant owners to be able to sign up for the service, so I can have more users and customers.

Complexity: S
Priority: High

Scalability

As a Huerate owner I want Huerate service to be stable enough to endure a lot of traffic, because I do not want to put a bad light on my service and discourage customers from using it.

Complexity: L
Priority: Medium

Multilingual interface

As a Huerate owner I want Huerate's interface to be available in as many languages as possible, because I want to reach to more potential customers.

Complexity: M
Priority: Medium

Marketing emails

As a Huerate owner I want to be able to send marketing emails to the restaurants in Czech Republic, because I want to convince them to use the service.

Complexity: L
Priority: Medium

Editing email templates

As a Huerate owner I want to be able to easily edit emails which are sent to users after various events such as registration, as well as marketing emails because I want to send to users the most up to date information.

Complexity: S
Priority: Low

Lead scoring

As a Huerate owner I want to be able to track activity on a site made by visitors and score it, so I can recognize interested visitors and turn them into users later.

Complexity: M
Priority: Low

Traffic analysis

As a Huerate owner I want to be able to analyze what operating systems and browsers do visitors use, because I want to know to which devices service has to be optimized.

Complexity: S
Priority: Low

Error reporting

As a Huerate owner I want to receive an email whenever error occurs on site, because I want to be able to fix it as soon as possible.

Complexity: S
Priority: Low

3.3.4 Backlog Summary

There are 28 requirements defined in this chapter with 129 story points in total. Eight of those requirements are marked as High priority and those requirements are essential in order to have a usable product. Requirements with High priority were developed as first. Requirements with Medium and Low priorities were considered as “nice-to-have” and few of them were not even implemented, because other more important requirements got attention. Chapter 4 and chapter 5 describe the design and implementation of the Huerate system based on the requirements listed in this chapter.

4 Design of Mobile Feedback Collection System

This chapter focuses on designing a system for the customer feedback collection – Huerate – from the technical point of view. Previous chapter enumerated requirements for the system. Those requirements were displayed as a snapshot from the end of a development phase. The design phase is approached in the same way. Apart from occasional remarks of what was done first, the design is presented also as it looked like at the very end, even if it was created incrementally.

One of the main requirements for the Huerate system is that it has to be very easy to use by restaurant guests on their smartphones. There are two ways how to run application on a smartphone. It can either be a native application written specially for the operating system which runs on a guest's smartphone, or it can be a web application which runs in a web browser. Native applications have to be installed through application stores which are present on all major mobile platforms. This installation can take a few minutes which is already more time than guests are willing to spend. The other disadvantage of native applications is that it has to be implemented separately for all mobile platforms. Because of this, Huerate is a web application optimized for smartphone browsers.

The design of the system's internal architecture and user interface design will be described in this chapter. Whole backend part of the system will run in the Windows Azure cloud. Choices which had to be made before developing the system for the cloud are described in section 4.1. The implementation of designed components and description how various technical problems were solved id described in chapter 4.7.3.

4.1 Running Huerate in the Cloud

There are several options, how to deploy a service to the cloud. In the last few years, it became very popular to run services and web applications in the cloud in order to save costs and reduce effort needed to maintain hardware and software of the web servers [13]. Running in the cloud means leaving maintenance of the certain layers of the infrastructure to the service provider. The basic level of the cloud service is Infrastructure as a Service (IaaS). In this level, service provider maintenances the whole hardware of the servers and user has access only to the operating system. In the IaaS model, user has to take care of software updates and of maintaining operating system in general.

The next level of the cloud offerings is called Platform as a Service (PaaS). In this model, service provider takes care of the hardware and the operating system, thus leaving on the user only the responsibility for running his application.

The lowest level is called Software as a Service (SaaS), where service provider takes responsibility for running the whole application and end user only uses it. The SaaS model is not suitable for development of the custom services or web applications, because it does not provide enough flexibility [14].

In general when deploying web applications, decision has to be made between IaaS and PaaS. IaaS offers greater flexibility, because it is possible to use any operating system with any custom software. On the other hand, it requires more effort to maintain it. In case of Huerate system, there are no special requirements for the underlying system or web server, so Platform as a Service will be used for the reason that it is easier to maintain.

Another decision which has to be made is which provider will be used to host a web application. Huerate system is an ASP.NET web application written in the C# language, thus a provider offering PaaS model with a support for ASP.NET has to be chosen. Microsoft's cloud offering called Windows Azure offers the needed configuration. In addition to that it is able to host a SQL Server inside Windows Azure and it comes with great integration with Visual Studio. Because of those reasons, Windows Azure was chosen to host Huerate.

Windows Azure's PaaS offering called Cloud Services is available in two variants [15].

- **Web Role** – deployed application of this type will act as a web application. The prepared virtual machine will have IIS installed and configured to run deployed application.
- **Worker Role** – this kind of Cloud Service is used to build services without web frontend. Usually APIs or services which have to compute a lot are deployed to this kind of Cloud Service. Service can have multiple public endpoints which can communicate with the outside.

4.2 System Architecture

Components of the Huerate system will be described in this section. The whole server side of the system is running in the cloud. Compute services are deployed to the servers in the cloud and data are saved in the cloud based database systems. Users access this infrastructure through web browsers. Diagram on Figure 4 shows components of the Huerate system and how they communicate with each other.

The majority of the work is done by the web application which communicates with the storage system and users' web browsers. This web application is deployed to the web farm[3]. There have to be at least two web servers in the web farm in order to eliminate downtimes caused by software updates and hardware maintenance which are done automatically by cloud hosting provider always at one server at a time. The web application is called WebUI and it is deployed to the Windows Azure Cloud Service as a Web Role.

Another compute service is the EmailSender. On contrary to the WebUI web application, EmailSender does not communicate directly with users. Its only goal is to send scheduled e-mails in the right time. It runs as the Worker Role type of Cloud Service and it is enough to have only one server of the service running at a time.

4.3 Web Backend

Web backend is the place where the majority of computing takes place. It is a web application called WebUI running in a web farm and handling requests sent by client's web browsers. WebUI serves three different user interfaces used by different roles.

Home

This part serves pages for visitors who visit landing page at http://huerate.cz. Goal of these pages is to describe restaurant owners what Huerate is about and convince them to sign up and try it for themselves. The graphic design has to be appealing to the eye.

[3] Web farm is a collection of web servers located behind the same IP address and running the same web application.

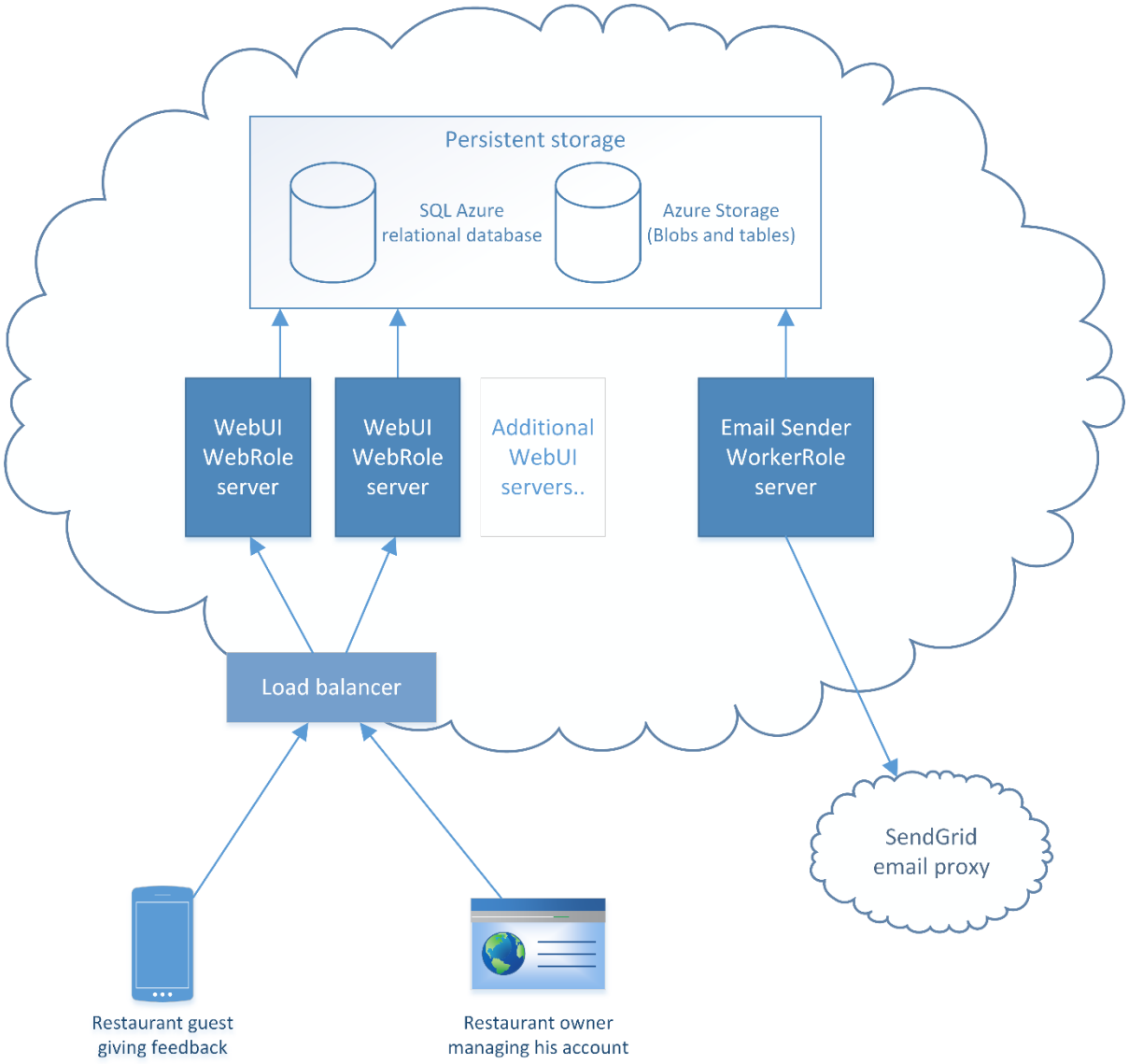


Figure 4: Diagram of Huerate's architecture

Feedback Form

Form is where the main focus is placed, because this is the place where the restaurant guests fill their opinions about the restaurant. This part of the web has to be optimized for smartphones and tablets. The other important factor here is performance. The guest will not want to wait for a page to load on his smartphone.

Administration

This part is used by restaurant owners who have already signed up for the service. Visiting this part is permitted only after signing in with a valid e-mail address and password which was used to sign up the restaurant. In this part, restaurant owners are able to configure their Feedback Form according to their needs as well as view feedback provided by their guests.

Web farm with the WebUI application is located behind the Windows Azure's load balancer which ensures that traffic coming from the outside is evenly distributed between all web servers.

Data such as cache cannot be stored on the web servers, because in the PaaS model which is used by the Huerate system the web server can be recycled in any time in order to perform software updates or other maintenance. Data stored on the web server's memory or file system are lost in that case.

4.4 Data Storage

The majority of data which has to be stored by the Huerate system is made by guests' answers and activity logs used for marketing purposes. Other than that, there are records about restaurants, their form definitions and translations.

Huerate system utilizes two storage engines for storing data. First one is relational database SQL Azure from Microsoft which stores relational data such as restaurant accounts, forms definitions and users' answers. Those information need to hold complex relationships and thus are better to be stored in a relational database.

The other type of storage engine used is Azure Storage. Azure Storage is a cloud based storage system which offers three different types of storage services which can be used to store large amounts of data and can scale very well.

4.4.1 Relational data

This section will describe schema of the SQL database used to store relational data in the Huerate system. Version of the SQL Server which is used is called SQL Azure. It is hosted in Microsoft's Azure datacenters and it is very convenient and secure to use it from Azure Cloud Services [16].

Huerate's database consists of 23 tables. The whole database schema is displayed in Appendix B. Main tables and relationships between them are described in this section.

RestaurantAccounts

RestaurantAccounts table stores records about restaurants registered in the Huerate service. It contains account information such as email and password which are used to log in, as well as information about the restaurant such as restaurant's name.

FormSteps

Every restaurant has its own unique Feedback Form which is used to collect feedback from restaurant guest. Form consists of form steps which can in turn consist of questions. There are 7 types of form steps available as described by requirement called Editing feedback form. Records in table FormSteps carry information which are common for all types of form steps such as to which restaurant this form step belongs and its order. In addition to that, there are another tables for each form step type carrying information specific to this type. For example table ConfirmationFormSteps, containing records for form type which displays the end message for the user, carries contents of text messages which are displayed.

Every record in a FormSteps table has to have a record in one of the tables for specific form step type. There is a foreign key relationship between FormSteps' Id column and Id column of every sub table. The relationships between the FormSteps table and other tables containing form steps can be seen on Figure 5 which displays a subset of the whole database tables.

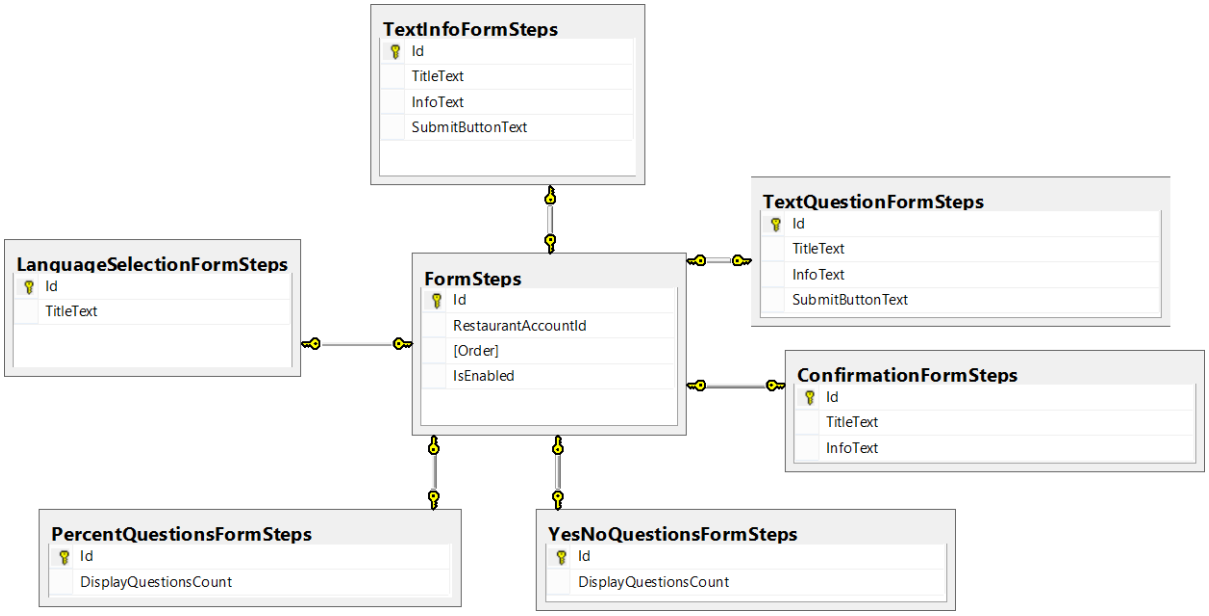


Figure 5: Relationships of the FormSteps table.

Questions

Questions table contains information about questions placed on the form steps. There are three types of questions available:

- **Percent question** – question which is displayed as a slider. This question stores information as a percent.
- **Yes/No question** – question with only two options: Yes and No.
- **Text question** – question which expects text answer.

Information about questions are stored in a tables structure similar to the one used for storing form steps. There is a parent table called Questions which stores common information and one additional table for each type of question. Tables are connected by their primary keys.

AnswerSets and Answers

Tables AnswerSets and Answers are used to store restaurant guests' answers to the restaurant's questions. Purpose of the AnswerSets table is to link answers to the individual questions together. Rows in AnswerSets table are identified by the Guid which is generated on the client side when user starts to answer questions.

Rows in table Answers represent individual answers to the questions. Structure here is organized in a similar way as in the form steps and questions case. There is a base table called Answers which carries common information such as time when answer was given and answer set to which the answer belongs. Then there are special tables for each type of question which carries link to the question as well as the answered value itself.

CustomTranslation

Finally there is a table carrying translations of the texts used in the form. Texts used on the form can be changed at the runtime, so they cannot be saved in the resource file. Details about how form translations are handled are in section 5.6.

ScheduledOutgoingEmails

ScheduledOutgoingEmails table is not connected to any of the previously mentioned ones. Rows in this table represent e-mails to be sent in a specific time. Those e-mails were used for marketing purposes. A fixed number of emails were scheduled to be sent each day and Email Sender Worker Role was retrieving them from the table and sending them. There is also a helper table called EmailTamplates which contains texts of the emails which are sent on various events. Apart from already mentioned marketing emails, there are templates sent after user signs up for the service or when user forgets his password.

4.4.2 Azure Storage

Other kind of persistent storage where Huerate system stores its data is Windows Azure Storage. Azure Storage is a cloud based storage system which offers two different types of storage services used by Huerate system [17].

Blob Storage

Blob (or Binary Large OBject) storage is an unstructured storage for large amounts of text or binary files. Blobs are organized into containers and can scale up to 100 terabytes. Blobs in containers can be accessed via REST API and it is possible to set public read access on a containers. This feature is used by Huerate system. It stores all of its public files, such as images, cascade style sheets and JavaScripts in a blob container and adds direct links to these files, so they are downloaded directly by browser, thus reducing load on the web server.

Table Storage

Table storage is another service offered by Azure Storage which is used by Huerate system. Table Storage is a NoSQL key-value database. It can store large amounts of non-relational data organized in columns. While Windows Azure Table Storage stores structured data without schemas, it does not provide any way to represent relationships between the data. Great advantage of this storage over SQL Azure is its price. It can store the same amount of data as SQL Azure for more than ten times smaller price [18].

Huerate stores debug logs in the Table storage. To reduce load on the SQL Server and decrease its size in order to pay less, another types of data could have been stored there. The best candidates are activity logs and scheduled emails, because they are large and do not require complex relationships. However, as SQL Server did not prove to be a bottleneck during testing, other requirements were more important and those data were left in the SQL Server.

4.4.3 Email Sender

In order to spread the word about the Huerate service and get users, marketing e-mails to restaurants were sent. EmailSender is a Worker Role Cloud Service which is responsible for sending those marketing emails. Delegating sending emails to the separate Cloud Service allows web servers to only focus on handling user requests and leaves work as sending emails to other service and thus not prolonging requests response time.

Email Sender's work is very straightforward. It checks the ScheduledOutgoingEmails table every five minutes and if there are emails to be sent right now, it retrieves them and sends them.

Emails are sent using the SendGrid email sending service. SendGrid is a cloud service able to deliver large amounts of email. It provides additional features such as tracking link clicks and mail opening. It is easier to use service link SendGrid than to maintain own SMTP server. With SendGrid's

LITE package, sending a thousand emails costs as low as $0.10 which was not a problem with the amount of emails sent. Communication with the SendGrid is made using their REST API [19].

4.5 Scalability

Requirement for the service scalability is reasonably high on the priorities list. It is important, because it is impossible to predict how big the load will be. Not scalable service suffers from increased response times when load increases and if it passes certain level, the whole service could become unavailable. This problem can seriously harm the growth of the service, as user who experience downtime at his first visit will unlikely come again.

If service is scalable, it means that it can accommodate its load capacity to the required traffic. Service has to be designed with scalability in mind from the very beginning. Scalability is about removing possible bottlenecks which can affect performance of the whole service. Possible bottlenecks will be analyzed in this section.

Not all solutions described in this section were actually implemented. The reason for that is that performance was not a problem during the testing and it was more important to implement more features in order to attract more users and know them better.

Web servers

Web servers are the place where most CPU intensive work is done. Web servers have to communicate with clients, parse requests, render HTML, communicate with storage engines, perform computations and so on. One server can handle only limited number of requests at time. The solution to this bottleneck is to allow servers to run in a web farm. This solution is much easier to implement when service is deployed in the cloud, because cloud hosting (IaaS and PaaS) provides web farm functionality out of the box. Service administrator can very easily scale the number of web servers to up to 20 servers[4] which in addition can be of a different sizes. However, the web application cannot use any kind of local cache, because changes made by different web servers would not be seen by others.

The other way of increasing possible load of web servers is to reduce amount of work the web server has to do. One practice is to move static files to the Content Delivery Network and thus do not burden a web server with serving files which do not require any kind of processing.

Another technique which can be successfully utilized is asynchronous programming. When web application is asynchronous, thread do not wait idle for some external resource and can be used to serve another request. When requests arrive, a free thread is taken from thread pool and assigned to handle the request. However, the number of threads in thread pool is limited and thus releasing thread when not needed can increase greatly the number of requests which can be handled [20].

And finally, one more way to increase load of web servers is to split its responsibilities into more independent web farms. In Huerate case, there could be separate web farms for Administration and for Form parts of the system. In extreme cases, even restaurants could have been split into separate web servers and thus providing virtually unlimited scalability in terms of computing power.

[4] With the Windows Azure Cloud Services [15].

Database

Database server is often a central part of the system where all data are stored and which all components use. This can cause problems of course. Relational databases have to ensure ACID[5] properties which make it hard to create farms of database servers.

A solution for having database as a bottleneck could be database partitioning which is a technique allowing dividing non related data into multiple servers. In Huerate case, that would be possible because individual restaurants are not related with each other and nothing is working with more than one restaurant at a time. With database partitioning it would be possible to put restaurants with a name begging with an A to N to one server and the rest to the second one for example. The distribution function can be modified based on the needs [21].

Another solution utilized to some extent in the Huerate system is to use non-relational databases which are not as robust as relational ones, but because of its simplicity can store large amounts of data and scale very well. NoSQL databases require complex queries to be made in memory rather than in the database server, so the implementation is often harder [17].

Caching the results of queries, which are expensive or done very often, is a practice which can greatly reduce amount of work SQL Server has to do. Distributed caching has to be utilized when application is running in web farm in order to propagate changes in cache between all web servers. Windows Azure provides distributed caching out of the box. Even the whole rendered output can be cached and thus greatly reducing load not only on SQL Server but also at the web server itself [22].

4.6 Security

Huerate service stores a sensitive data which can harm restaurant if leaked to public. Also if some security vulnerabilities are exploited, it can seriously harm service's reputation. There is a requirement for securing the service called Secured form results.

The most important thing to secure is user's account, because if the account is compromised, anybody can access restaurant's answers. Account is secured by storing its password in a salted hashed form. A salt is a 256bit number unique for every restaurant. Salt is stored in the table with the restaurant and is appended to the plain text password before applying hash function. Using salt makes it practically impossible to use rainbow table to retrieve plain text password from the hash. A SHA-2 with the 256bit output hash function is used to hash password before storing it in the database [23].

Another security enhancement, which would make getting access to somebody else's answer more difficult, is encrypting HTTP communication especially when signing in or visiting administration. Using unencrypted HTTP when entering credentials is considered to be very insecure because if user is on an open wireless network, anybody on the same network can eavesdrop them.

Database's security is enhanced by using a firewall which allows connection only from the web server's IP Address. In any case, data stored in database could be encrypted as well.

Naturally, web application must not be vulnerable to general attacks such as Cross Site Scripting and SQL Injection. Those attacks typically make all previously mentioned security enhancements useless because attacker can bypass it.

[5] Atomicity, Consistency, Isolation, Durability - set of properties that guarantee that database transactions are processed reliably

4.7 User Interface Design

This section show how various user interfaces used across the system were designed. As described in chapter 4, there are three parts of the system:

- **Feedback Form** – part where restaurant guests provide feedback on their mobile devices.
- **Administration** – part where the restaurant owner checks feedback on his restaurant and manages his Feedback Form**.**
- **Home page** – part for the restaurant owners who are looking for feedback collection system.

Each part serves different purpose and is used by other kind of users. Design of the user interface will be done with those aspects in mind.

4.7.1 Feedback Form

This is the most important frontend part of the system. If this interface is not done correctly, it will discourage restaurant guests from filling in the form and thus the system will not meet its goal. Interface is designed as multi step wizard, so that user can focus only on one type of questions at the time. The sequence of form steps which will be displayed is editable by restaurant owner. The most important steps will be shown in this section. A Form's appearance on a table device will be shown. When opening a page on a device with smaller screen, the left image will be removed and the whole page will shrink.

First design displayed on Figure 6 shows how step with only one percent question looks like. By default, there is a form step of this type asking for guest's overall satisfaction with the visit.

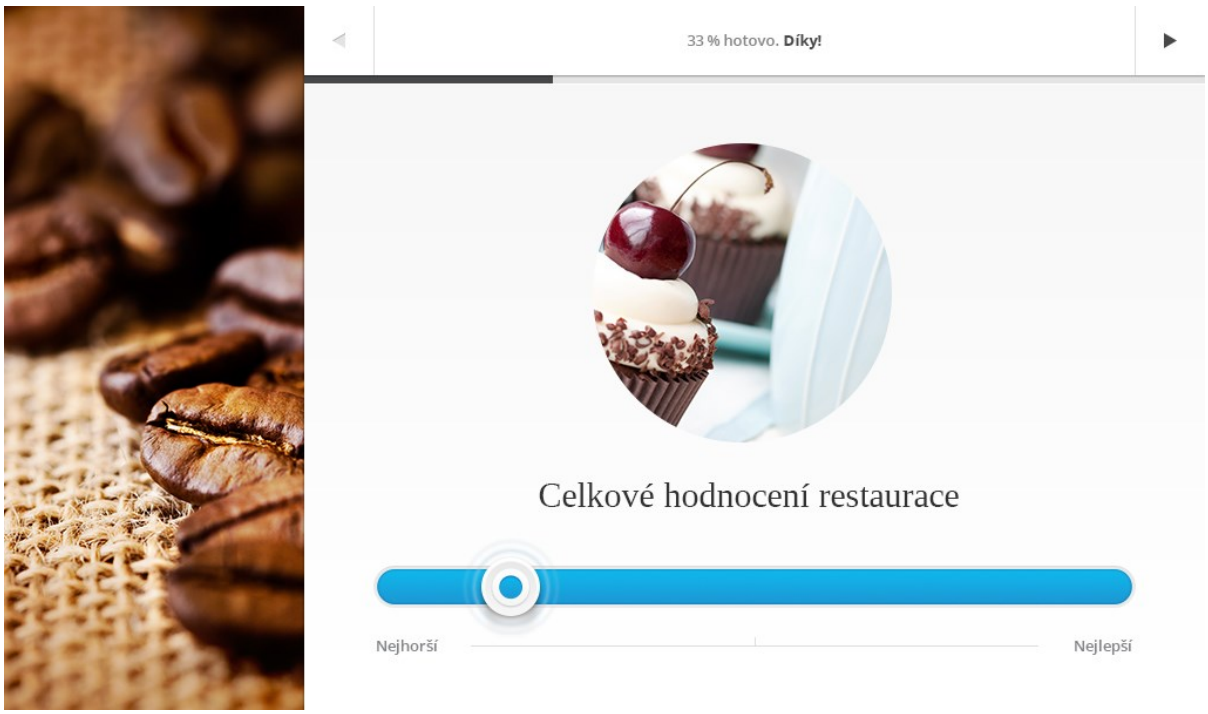


Figure 6: Form step with question asking for overall satisfaction with the visit

When adding more percent questions to the form steps, the sliders and images will become smaller, so up to three questions will fit the screen. By default, there is a form step with three percent questions asking for opinions about Food, Service and Environment. Those three areas were chosen as most

important by interviewing restaurant owners. Figure 7 shows how a form step with three percent questions looks like.

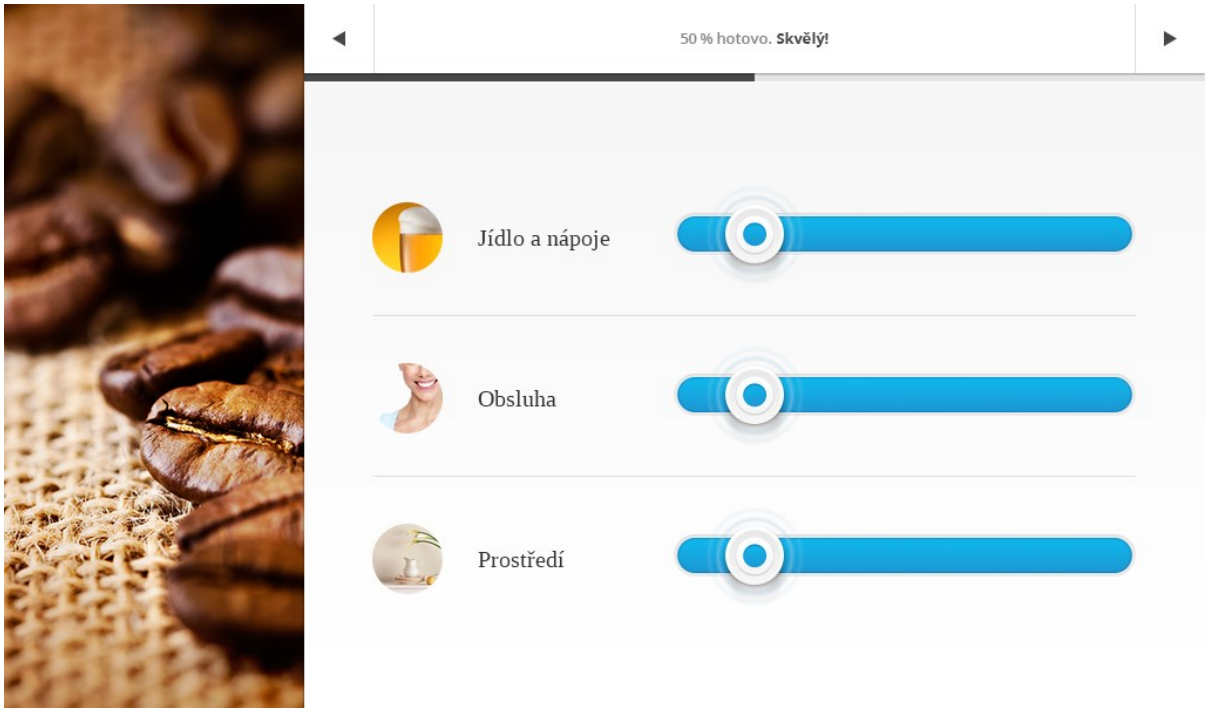


Figure 7: Form step with three percent questions

All opinions cannot be expressed using sliders and Yes/No questions, so there is additional form step type available – a text question form step. Figure 8 shows how a form step with question expecting answer looks like.

Figure 8: Form step with a question expecting text answer

4.7.2 Administration

Administration is where the restaurant owners check answers given by their guests and manage their restaurant. It will be possible to view results from the individual days. Figure 9 shows early design of the administration user interface. During the implementation phase, the administration user interface changed a little based on the results of user testing. Paging of text answers was removed and all text answers from a given day are displayed.

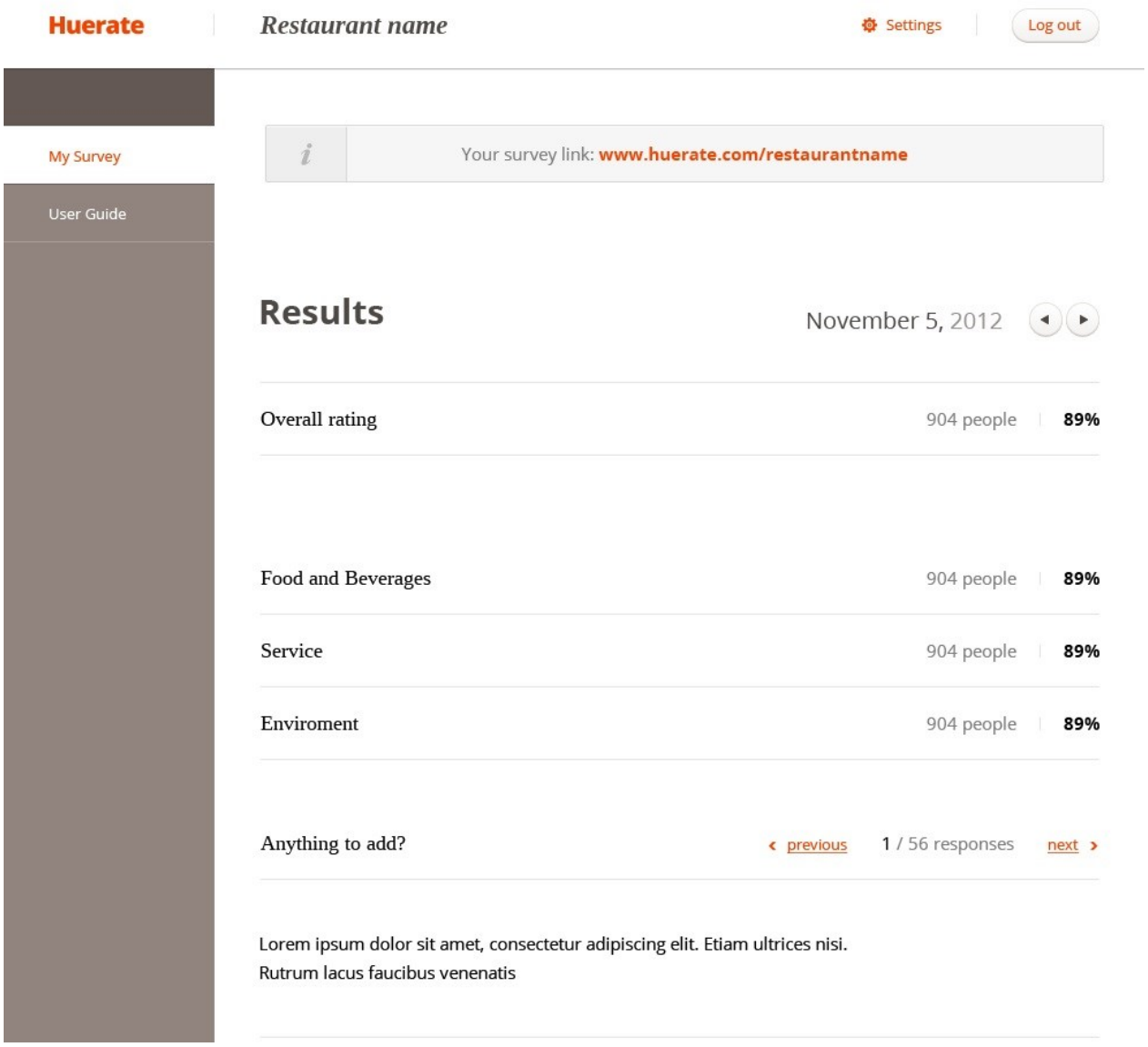


Figure 9: Prototype of the administrative interface

4.7.3 Home Page

Landing page is where visitors looking for a feedback solutions will come. The focus is put on a clean and nice design, short service's purpose explanation and a call to action, which is restaurant registration. There is also a login button for the visitors who already registered their restaurants. There is also a form with a possibility to subscribe to a newsletter with news about the service at the end of the page. The landing page is available in Czech and in English. The Czech version is located at http://huerate.cz and

the English version is at http://huerate.com. The localization system of the Huerate service makes it very easy to add new languages.

Figure 10: User interface of the landing page in English

5 Implementation

This chapter describes how components designed in chapter 4 are implemented. Previous chapter lays down foundation of the system architecture, but does not go into details about how individual parts of the system are implemented. Tools and libraries which were used during the development are also briefly introduced in this chapter.

The whole system is implemented as a Visual Studio solution. The web frontend is an ASP.NET MVC web application with C# and JavaScript programming languages used on the server side and client side respectively. Underlying parts of the system, such as data access layer or services layer are split into separate projects of Class Library type in order to divide responsibilities into separate assemblies and to make them available also for other applications which are used by the system.

There are two projects in the solution responsible for deploying cloud services into a Windows Azure cloud, one for web frontend and the other one for email sender. Those projects holds configuration of the cloud services and code which is executed after services are deployed.

In the separate solution folder called HelperApplications console or desktop applications used for service maintenance are located.

5.1 Responsibilities Distribution between Assemblies

The web application is split into many assemblies in order to properly distribute responsibilities into separate parts. Those assemblies create layered architecture where higher parts of the solution depend on the lower parts, but not the other way around.

All underlying assemblies are represented by Class Library projects in Visual Studio. This kind of project compiles into a dynamic-link library (dll) which is later referenced by the higher layers. The highest assembly is represented by ASP.NET MVC project. All assemblies and their dependencies are shown on Figure 11.

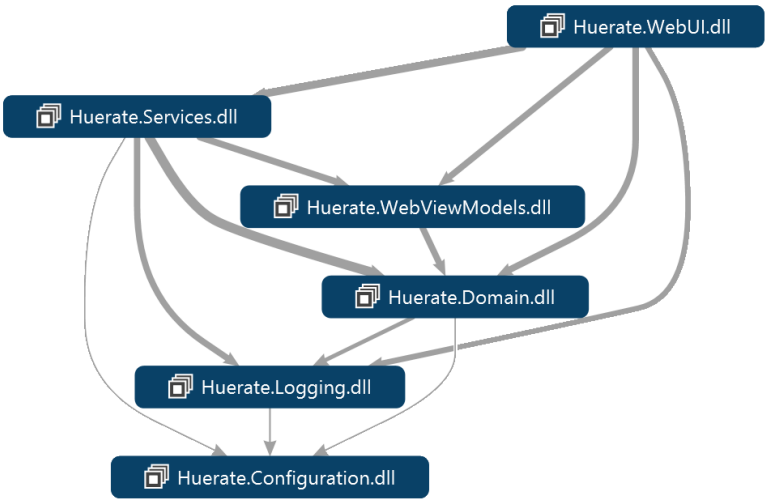


Figure 11: Assemblies and their dependencies

The very bottom assembly is **Huerate.Configuration** which is a very simple assembly whose only responsibility is to read settings and provide them to the rest of the solution. **Huerate.Configuration** automatically recognizes if application is running inside the Windows Azure web role and reads settings from the correct configuration file – ServiceConfiguration.cscfg if running inside Windows Azure and web.config otherwise. This assembly does not depend on any other part of the solution.

Second from the bottom, there is an assembly called **Huerate.Logging** capable of writing log messages into a variety of destinations. Using this assembly, it is possible to log different levels of messages from debug ones to fatal errors. There is a LogService class exposed to the outside, which contains logging methods. This class uses an external library log4net under the hood, but does not require code which uses it to reference log4net. Log4net is a library capable of outputting log messages to the variety of output targets. It is a .NET port of the log4j Java logging library [24]. **Huerate.Logging** allows logging into files, console or Azure Table Storage. **Huerate.Logging** uses **Huearte.Configuration** to retrieve settings needed for logging, such as files names or Azure Storage connection string.

Next comes the data access layer represented by the **Huerate.Domain** assembly. This assembly is a bridge between persistent data storage and the rest of the application. Its detailed functioning is described in section 0.

Above data access layer lays **Huerate.Services** assembly which is where the most of the application logic happens. This assembly contains services grouped by their purpose. The services layer is described in detail in section 0.

On the very top lays **Huerate.WebUI** assembly which is an ASP.NET MVC application itself. This project is responsible for communication with the user using his web browser. The result HTML pages are rendered in this assembly. The details of this project are described in a separate section below.

5.2 ASP.NET MVC Application

The web application itself is located in a project called **Huerate.WebUI**. It is the highest project in a dependency chain and it references all other projects. The web project is of ASP.NET MVC web application type. ASP.NET MVC is a web application framework developed by Microsoft. It is part of the ASP.NET framework and shares a lot of functionality with the older Web Forms framework [25]. MVC is a design pattern whose main goal is to separate applications' logic from the user interface. MVC stands for Model-View-Controller where Model represents a state, data and logic of the application, View represents user interface and Controllers handles interaction with user. Controller parses input parameters and calls appropriate model methods and view methods [26].

The majority of the controllers located in the WebUI project are responsible for handling GET requests sent by user's browser when he wants to display a web page. An ASP.NET MVC framework knows what controller to call based on a set of defined routes. There is a route for every controller as well as default route which serves a form for a restaurant. Some of the controllers contain only API methods which are called through AJAX. An example of this type of controller is AnswerController which handles saving user votes.

According to the design of the web backend in section 4.3, the web application is split into three parts with different user interface, target audience and primary goal:

- **Home** – landing page which can be visited by potential users
- **Form** – mobile optimized web used by restaurant guests to provide feedback
- **Administration** – place where owners of the restaurants can manage their forms

There is a feature in ASP.NET MVC called Areas, used specifically for splitting one application into more unrelated parts. Using Areas, it is possible to divide application into absolutely independent parts (called Areas) and make route engine resolve which Area should be used at which time [27]. However, in this case different parts will share a lot of common functionality and content (images, styles, JavaScript files), so only Views will be split into a different folders and the rest will be kept together.

Huerate system uses client side scripting when the user interface needs to be responsive and refreshing the whole page after every interaction is unacceptable. The best example of this approach is the Feedback Form page itself. The whole process of responding to the questions happens without a single page refresh and answers are sent to the server through AJAX. Connection speed and browsers in general are slower on mobile devices, so limiting data transfer and rendering is desirable. Knockout JavaScript framework is used to simplify client side development. Knockout is an open source framework able to bind data to the user interface and therefore eliminate the need to manually synchronize data and UI [28]. The Knockout library is utilized mainly in the Form page and in the form editing interface in administration. The whole editing is made on the client side and only raw data are sent to the server to be saved.

Sass (Syntactically Awesome Stylesheets) language is used to simplify CSS styles creation. Sass is a scripting language which is converted to CSS before publishing a web application. Using Sass it is possible to reuse elements styles, use variables for common metrics and cascade rules by putting them into one another [29]. Newer SCSS syntax of Sass is used. All CSS files in the Huerate project are written using Sass. Mindscape Web Workbench is a Visual Studio add-on which was used to write Sass styles. Conversion from Sass to CSS is completely transparent using this tool [30].

WebUI does not send entities straight to views to be rendered. Instead, it converts data from entities to Data Transfer Objects and send them to views. Data Transfer Objects (DTO) are objects whose only purpose is to carry data, they have only properties and no methods. DTOs are located in a separate project **Huerate.WebViewModels**. This project is referenced by both WebUI and Services, so services can serve directly DTOs. Where conversion from entities to DTOs is complex due to large number of properties, a free library AutoMapper is used. AutoMapper can automatically copy data from properties in one object to properties in another. Properties names must match or custom mapping have to be specified [31].

5.3 Domain Layer and Entity Framework

Domain layer is the place where all database access is done. It is implemented in the **Huerate.Domain** project. Domain project lays low in the hierarchy; it references only Logging and Configuration assemblies. All access to the database is done using object-relational mapping and no plain SQL query is used. Using this approach it would be possible to switch database engine from SQL Server to another one in the future. An ADO.NET Entity Framework is an object-relational mapping framework for .NET framework. Entity Framework maps database tables to the .NET classes called entities. This mapping is done based on an Entity data model, which is a set of rules defining how to map tables to classes. Entity data model can be defined manually in an .EDM file or created dynamically on runtime by Entity framework. The latter option, called Code First, is used in a Huerate system. Using this approach, Entity framework reads structure of the entity classes and the conventions specified in a configuration file. Using those information, model is created in memory [32].

Entities in a Domain projects corresponds to the database structure designed in the chapter 4. There is one entity for each database table and its name is the same as database table name, but in singular.

Even though Entity Framework is used to access database, the classes which use Domain assemble do not have to know about it. No Entity Framework type is exposed outside of the Domain Project. Because Code First approach is used, entity classes do not derive from any common base class. The queries done using Entity Framework context objects are encapsulated into repositories. Each repository represents access to one entity. There is also a UnitOfWork class which provides access to all repositories and has method Save which saves all changes done to entities retrieved by this instance of UnitOfWork. Every repository contains common methods for getting entity by id, getting all entities or adding or deleting entities. Those methods are implemented in a common generic base repository class. If some entity needs a specialized method for accessing data, it has its own repository class which implements those methods.

Database schema was changed over time as new features were added. Entity Framework has tool called Migrations which enables developers to work with different versions of database schema. This tool was used during the development to automatically move database schema to the newest version. Entity Framework Migrations automatically recognizes when model changes and generates a class deriving from DbMigration. Class deriving from DbMigration represent one version of the database schema and contains methods called Up and Down for moving schema between this level and one level below [33]. Classes representing database schema versions are located in folder Migrations in **Huerate.Domain** project.

5.4 Services Layer

Project **Huerate.Services** takes care of all application logic of the Huerate system. This project contains a number of services. Service is a class responsible for a specific task. Each service consists of an interface and actual implementation. This design allows code to be unit tested as well as use a dependency injection pattern which is described in detail in section 5.5. Services from this assembly are generally usable even if the application is not running in an IIS server. The only exception is UrlService which needs to know how to build routes to individual pages.

Services are divided into two groups based on their purpose:

Data services

Data services extend functionality of the repositories from the Domain Layer. Those services are required when manipulation with multiple entities is needed, because repositories can only work with one type of entities. Also, conversions from entities to models used in views are done in Data services. All data services extend DataServiceBase class and implements IDataService interface or its descendant.

Functional services

Functional services provide certain functionality to the rest of the solution. They are not linked with any entity. Examples of Functional services:

- **DefaultDataService** – contains default data, such as questions which are created for new restaurant
- **SendGridEmailSenderService, ScheduledEmailSenderService, SmtpEmailSenderService** – three services which are able to send emails using SendGrid or directly through SMTP server. **ScheduledEmailSenderService** is able to send scheduled emails and is used in the Email Sender worker role.

- **QrCodeService** – service able to render QR codes
- **MembershipService** – handles user authentication

Services project takes care of localization of the user interface as well. There is a special service called TranslationService which acts both as Data and Functional service. This service is able to translate given resource string into a requested language. Static translations which are used to localize user interface (mainly Home and Administration parts) are stored in .resx resource files. There is also second kind of translations which are used when localizing Forms. Translations in the Forms are changed on the go by restaurant owners, so .resx files cannot be used, as they have to be compiled after every change. Dynamic translations are thus stored in the database and this is where Data service nature of TranslationService is. Details about Forms localization are described in section 5.6.

5.5 Dependency Injection

Dependencies on concrete types are not hardcoded in the whole Huerate system. Instead, all classes depend only on interfaces which represent contracts, but do not imply how contracts are implemented. This applies mainly to the services from Service Layer and Controllers in WebUI. Those classes accept interfaces in their constructors and concrete implementations of those interfaces are resolved at runtime by a component called Dependency Injector. In fact, Services Layer exposes only Dependency Injector object and available interfaces. Concrete implementations of the interfaces are hidden inside the assembly by marking them as internal.

Ninject open source library is used as a Dependency Injector in the Huerate project. It is very simple to define bindings from interfaces to concrete implementations using Ninject and apply them on new classes. Ninject is able to resolve dependencies recursively – if class which is needed needs implementation of another interface, it is also resolved [34].

Dependency Injection improves code quality if applied correctly. By not hardcoding dependencies, classes become loose-coupled and highly-cohesive. Also, changing implementation of any component is very easy later in the development life cycle. Relying on interfaces instead of classes is one of the most important requirements needed, in order to make code unit-testable. By relying on interfaces, it is possible to inject mock into the tested class in the unit test, and thus test only one part of the code base [35].

Code snipped below illustrates how a classes with and without hardcoded dependencies look like.

```
// Class which depends only on contract specified by IQrCodeService
public class ImageControllerDi : Controller
{
    private IQrCodeService QrCodeService { get; set; }
    public ImageControllerDi(IQrCodeService qrCodeService)
    {
        QrCodeService = qrCodeService;
    }

    public FileResult QrCode(string restaurantCodeName)
    {
        return QrCodeService.GetQrCode(restaurantCodeName);
    }
```

```
}

// Class where dependency on QrCodeService is hardcoded
public class ImageControllerHc : Controller
{
    public FileResult QrCode(string restaurantCodeName)
    {
        var qrCodeService = new QrCodeService();

        return qrCodeService.GetQrCode(restaurantCodeName);
    }
}
```

ASP.NET MVC web application uses custom factory provided by Ninject to create controllers. Using this approach, even controllers benefit from using dependency injection.

5.6 Multilingual Forms

A requirement, which was very high on a priorities list, was the ability to create forms available in multiple languages. This requirement was actually moved higher on the priorities list, because a lot of restaurant owners stated that the product is absolutely unusable for them if the feature is not present. The reasoning behind this is that they have only one set of table material and it has to be multilingual. They cannot dare to lead their guests from abroad to a survey which is in Czech.

As was said in the section about Services Layer, custom translations cannot be stored in the .resx files, because they have to be recompiled every time translation changes. Another problem which has to be solved is the fact that translations from one restaurant have to be available on the client side, because user will be able to select his language in the Form web page and it will take effect without refreshing the whole page.

Therefore, custom solution which solves both problems has been implemented. Translations are stored in the persistent database, specifically SQL Server. There is a special Table for translations called CustomTranslations. This table stores mapping between resource strings and texts in different languages. Every translation is assigned to restaurant which owns it. Translations are made available on the client side by rendering it in the multidimensional associative JavaScript array and including it into a page. After changing language, values from different array key are taken and rendered values are replaced with the new ones by the Knockout library.

5.7 Helper Applications

Huerate solution contains a number of helper applications which take care of the maintenance of the service. Those helper applications are generally single-purpose applications, so they are not configurable and their user interface is very simple. Also, the code quality is not as high as in the other parts of the system, because those applications are for internal use only, so dangers from running the buggy code are lower. There are these helper applications available:

DatabaseDataTransfer

This is a console application which connects to the two Huerate databases in different versions and transfers data from older version to the newer one. This application was needed when bigger change to the database which could not be handled by Entitiy Framework Migrations was introduced.

EmailSender

Console application which runs in a loop and sends all scheduled emails.

EmailScheduler

Application capable of scheduling configurable number of marketing emails to be sent on the next morning. This application was used to uniformly distribute sending of the marketing emails.

EmailTemplateEditor

This is a desktop application written in WPF which is used to edit email templates. Email templates are stored in the databases, so it would be very inconvenient to edit them by hand.

BlobUploader

Console application which uploads all web objects (css files, JavaScripts, images, etc.) to the Azure Blob storage, so it can be later referenced by the website. This application has to be run whenever the web objects change.

6 Deployment, Testing and Evaluation

The whole development phase of the Huerate system took over a year, but with a long pause after developing the first version. At first, the goal of the Huerate project was not only this thesis but also a standalone service offered commercially. The principle was different back then, system was meant to be used on tablets located in the restaurants. The MVP which consisted of simple Form frontend, administration wireframes and simple home page was developed by the team of three. The MVP was presented to the restaurant owners in Brno, but without any success. Restaurants owners' main problem with the idea was the cost of the tablet devices which they would have to acquire. Because of the negative reception from the restaurant owners, the project was discontinued and the development team fell apart.

After three months pause and eight months before the thesis deadline, the original idea was changed so that the feedback will be given on the guests' smartphones, as it is described in this thesis. Development of the new system began again but now only by the author of this thesis. Foundations of the original system were kept and development of the new one was continued on top of that.

The new Huerate system was developed using simplified Scrum methodology as described in section 3.1. Scrum development is based on short iterations called sprints after which the project should be functional and the progress should be consulted with users. This process was met and Huerate was deployed and presented to users from the very beginning. Communication with early adopters about features implemented so far has proven to be a great help with ensuring that project is heading in the right direction. Many requirements which were considered as important at first were later moved lower on the priorities list based on the feedback of early adopters.

First section in this chapter describes the process of deploying the application after each sprint. Second section explains the role early adopters played in the evolution of the idea and how were they convinced to take part in the project. Finally the reception from the restaurant owners and guests is summarized and the overall usage of the Huerate system in real live is assessed.

6.1 Deployment to the Production Environment

Development of the Huerate system took eight months. This time was divided into eight 4-weeks sprints and after each of them the new version of the system was deployed to the Windows Azure cloud in order to be able to get early feedback.

Windows Azure Subscription had to be acquired before deploying system. Fortunately, Microsoft runs a program called BizSpark which entitles young companies and standalone developers to use Microsoft software and use its services for three years for free [36]. Members of the BizSpark program receive a Windows Azure Subscription which allows them to run reasonably big services with sufficient amount of data storage[6]. Huerate's application to the BizSpark program was accepted and thus it could be deployed without any costs. Also Visual Studio 2012 Ultimate which is free to use by the BizSpark members was used to develop the whole system.

Web application – the core part of the Huerate system – was deployed as a Web Role. Small server size[7] was chosen and number of server instances was set to two. Every cloud service can be deployed to the two environments: a **staging environment** in which the application can be tested before

[6]Details about the offer can be found at this address http://www.windowsazure.com/en-us/offers/ms-azr-0012p
[7]Small web server is a server with 1 dedicated CPU and 1,75GB of RAM [38]

it is promoted to the **production environment**. Both of the available environments have their own DNS entry and they can be fully used simultaneously.

New version of Huerate system was always deployed to the staging environment first and then tested briefly if everything was running correctly. After that, the VIP[8] Swap operation was performed. This operation switches almost immediately production and staging environments, so there is no downtime.

The management of the all Windows Azure components can be done through the Azure management portal located at http://manage.windowsazure.com. Separate Storage service accounts used for storing log files and serving web objects to users as well as SQL Azure database were created through the Azure management portal. After setting up all required services in the management portal, the application can be published straight from the Visual Studio 2012.

Number of server instances can be set even after deploying the service. This option was used to test if application behaves correctly when running on variable number of instances. Figure 12 shows how setting number of server instances looks like in the Azure management portal.

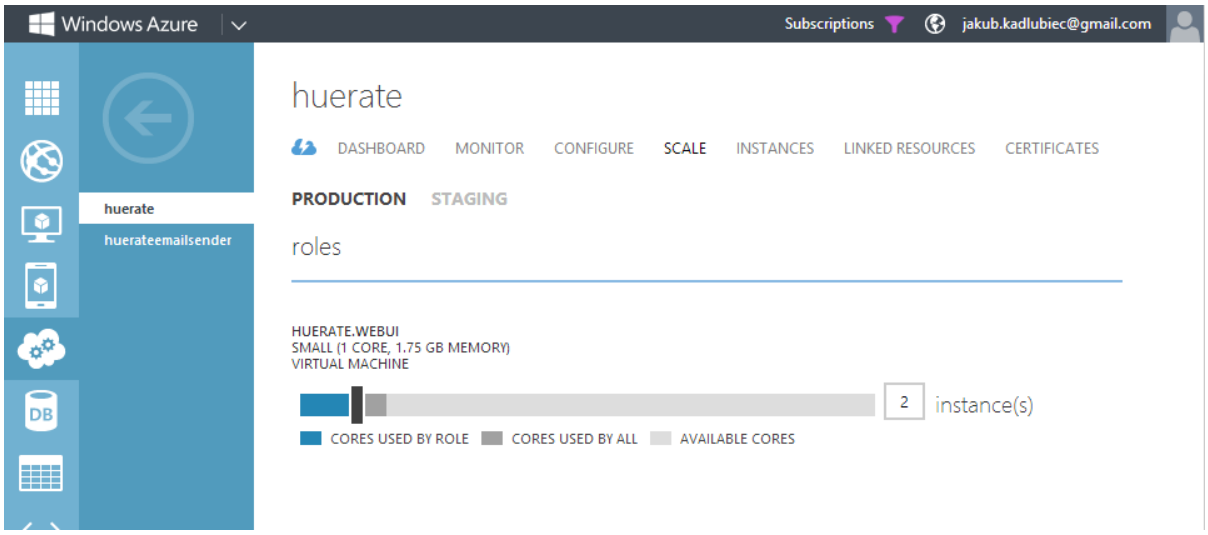


Figure 12: Setting number of server instances in the Azure management portal

Deploying system is not enough to get feedback. Early adopters who are willing to use system even if it is not fully functional are needed. Next section describes how those early adopters were acquired.

6.2 Early Adopters Acquisition

Early adopters are users of a product which is not yet used by the general public. The product often does not have to be fully developed and thus it is not known how it would perform and if it would work as expected. Early adopters are willing to be exposed to risks and problems of using a product with a lot of bugs. In exchange to that, they receive special attention from the product vendor and access to the technology which is not yet common in the majority of the market [37].

Acquiring early adopters of the Huerate system was crucial, as project author does not possess any experiences with running a restaurants and with the restaurant business overall. Early adopters thus provided valuable insight into this environment.

[8] Virtual IP address

Majority of Huerate's early adopters were acquired by direct e-mail sent to the owners of the restaurants. E-mail addresses were extracted from the public profiles of the restaurants located on the two popular websites lunchtime.cz and restaurace.cz. A small web crawler which was able to extract all e-mail address from the two mentioned websites was written. The crawler was able to get all e-mail addresses from both sites in less than an hour. At total, 15 710 restaurant profiles were extracted, but many of them contained corrupted or duplicate e-mail addresses. Detailed statistics of sent e-mails are presented further in this section.

Message to the restaurant owners briefly summarized what the project is about and what problems does it solve and most importantly asked them for collaboration in two areas:

- **Survey** – restaurant owners were asked to fill in information about their restaurant into a survey. The goal of the survey was to know how restaurant owners collect feedback currently better and to recognize what they need from a system for feedback collection.
- **Sign up for the service** – restaurant owners were asked to become early adopters and start using Huerate in their restaurants.

E-mails were sent in batches in order to not discourage all potential users by presenting them the very first version. E-mail campaign resulted in 89 accounts created and 17 actual early adopters acquired. To be counted as early adopter, the restaurant had to have at least 5 guests filled the survey in the last two weeks. The diagram on Figure 12 illustrates the process of acquiring early adopters.

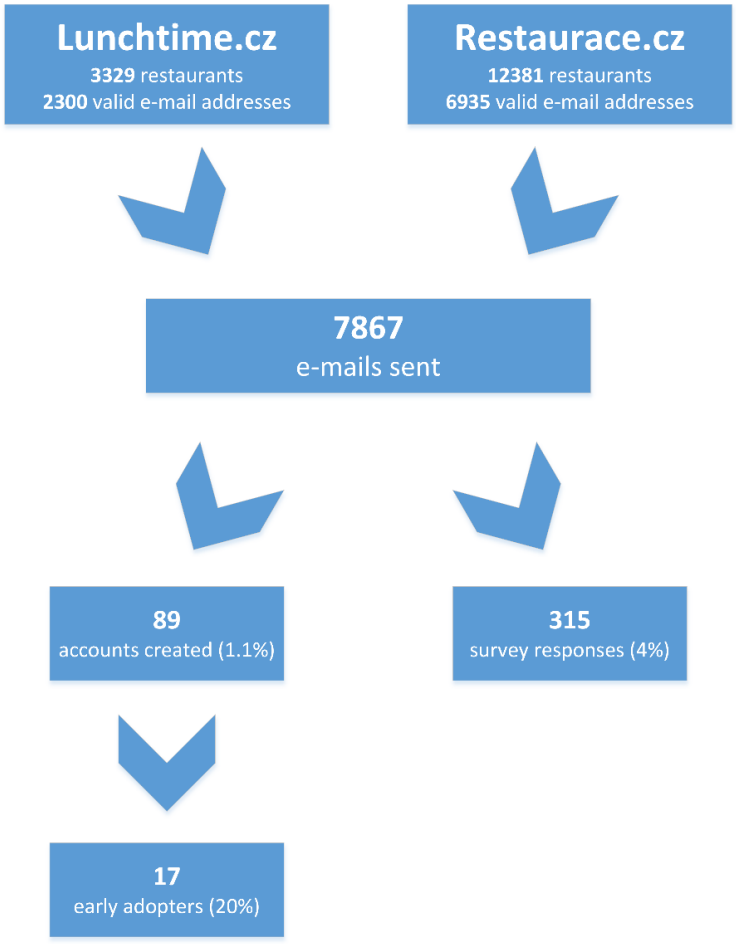


Figure 13: E-mail campaign

Even if number of valid e-mails extracted from Restaurace.cz was almost three times higher as number of e-mails extracted from Lunchtime.cz, the actual number of accounts created was almost the same for both of the sources. The reason is that Restaurace.cz has a lot less up to data database and a lot of e-mails got from Restaurace.cz simply did not exist at the time of writing.

The content of the marketing e-mail message can be found in Appendix C.

In addition to the one time e-mail campaign, there were a lot of individual e-mails sent between restaurant owners and Huerate author. To be specific, 173 e-mails were received from restaurant owners and 86 were sent back to them. Incoming emails contained mostly the information that the restaurant does not exist anymore. In addition to e-mail communication, also 6 personal meetings with restaurant owners were arranged and about 25 phone calls were made.

6.3 Usage at the Restaurants and Feedback from Users

This section describes what steps had to be done in order to start receiving feedback from the restaurant guests and what problems had to be solved on the way. After the successful deploy, restaurant owners and guests who were providing answers were asked about their experience. Their opinions as well as results of the user testing of the user interface are summarized at the end of the section.

After signing up for the service, restaurant owners had to give their unique Feedback Form link to their guests. The way how this step was done had a great impact on the number of responses the restaurant received. Some of the restaurants simply published the link on their website or Facebook page. This usually meant getting a lot of responses immediately, but after short time the responses stopped coming. This way of publishing link breaks the purpose of the service, because it is meant to be used right after visiting the restaurant when the impression is fresh. All restaurant owners were encouraged to put printed notices with the link at the tables at their restaurants. This way of publishing link meant a stable growth of the number of responses received at a pace close to 4 responses per day on average. Figure 15 and Figure 16 located in Appendix D show examples of the flyers which were used by restaurants to lead guests to their Forms.

A lot of restaurant owners did not want to put a flyers on their tables, because they like to keep tables as empty as possible. Another concern of both restaurant owners and guests were the Huerate name. They were not able to pronounce it correctly and the name did not mean anything to them.

Restaurant owners and guests were interviewed and their opinions were noted down. Interviews did not have the same structure and opinions were different a lot of times, so only the most frequent ones will be presented.

When working with restaurant owners, it became apparent that they need as much assistance with setting up the service in their restaurants as possible. Based on that discovery, a help page describing the whole process step by step was written and sample flyer was made. According to restaurant owners it proved to be a great help to them.

The biggest advantage of the service which was mentioned by almost all interviewed subjects was its simplicity and ease of use. Restaurant owners said that they received offers for similar services before, but they were so complicated that they were not able to use them. Guests appreciated that answering the questions did not take a lot of time. Owners who used flyers on the tables were mostly pleased by the amount and quality of feedback they were receiving. One restaurant owner has changed one room of his restaurant to non-smoking after using Huerate for a month. He did not know about the problem before, but his guests expressed their displeasure very early in the Huerate form.

6.3.1 User Testing

In addition to verbal interview, a user testing with restaurant owners and guests was made. Three restaurant owners and ten guests were tested. A user testing is a technique where real users use the tested systems on their own and an evaluator watches what actions they take and what problems do they encounter. User testing can reveal a lot of problems with the user interface which were not discovered by the author, because he designed it himself and thus it looks intuitive to him [38].

User Testing with Restaurant Guests

Guests were tested during their actual visits to restaurants. They were asked to use their mobile devices to navigate to the Feedback Form and to fill it.

All of the tested subjects were able to correctly navigate to the Feedback Form. Majority of them used QR code. All guests understood the user interface of the Feedback Form and knew what was asked of them on every step. The issues arose with controlling the Form's slider mainly on older phones with Android operating system. The experience with the slider was limited because of the low performance of those phones. Moving slider's handle was not fluid and because the Form sometimes moved to the next step before finishing the slide, it was impossible to provide an accurate answer.

Another problem was with the arrows used to navigate between form steps. Around half of the users did not find them and thought that the answers on questions from previous steps cannot be changed.

User Testing with Restaurant Owners

Three restaurant owners participated in the user testing of the administration interface. One user testing session consisted of three tasks:

1. Sign in to the administration interface and check guests' answers from the previous day
2. Change name of the restaurant
3. Edit form (all possibilities were tested)

Task number one did not cause any difficulties for the restaurant owners. It took them few seconds to locate the button used to switch days.

Changing the name of the restaurant was also successfully finished by all tested restaurant owners. They did not navigate straight to the correct menu item but instead went through all pages with "editing" or "settings" in its name. When they got to the right page, it was easy to find out what to do.

The third task which tested the ability to customize Feedback Form revealed some usability issues. Restaurant owners were confused by the languages editing interface. They did not understand the difference between buttons "Add" and "Save". When editing texts on the form steps, restaurant owners thought that the saving button saves texts in all languages they changed. The purpose of the field "Number of displayed questions" was not clear. Also, the whole concept of form steps was confusing to the users and they did not understand their purpose and what do individual steps do. Eventually, all restaurant owners were able to achieve specified tasks but the experience could have been better.

.

7 Conclusion and Future Work

The idea behind this thesis was to create a system for restaurant owners which would help them get more feedback from their guests and thus allow them to improve their service in the most important areas. The problem of not getting enough feedback proved to be real because according to the survey made among 315 restaurant owners, only 34% of them are satisfied with the amount and quality of feedback they are getting.

It was impossible for the author of this thesis to come up with the complete specifications for the Huerate system because of his lack of experience with the restaurant business. Because of that, it was essential to collaborate with the restaurant owners. After developing a product with the minimal set of features and publishing it at http://hureate.cz, restaurant owners were approached and asked for collaboration. Even though they admitted they would like to get more feedback, it was hard to convince them to try out the new system. Restaurant owners were conservative and reluctant to trying new things. Communication with them was very difficult and time consuming and this problem prevailed until the end of the development phase. Nevertheless, five restaurants agreed to collaborate from the very beginning and helped to shape the Huerate system into its final stage.

The e-mail campaign was launched after developing more features with the goal of getting new users. 7 867 restaurant owners from the Czech Republic were addressed with the proposition of using the system for free in their restaurants. Almost 100 restaurants signed up for the system but only 17 of them used it actively. This number is considered to be a disappointment, because it makes only 0.2% of all addressed restaurants. Still, those 17 restaurants were actively asking their guests for feedback and were getting answers from four people per day on average.

After having 17 early adopters it become challenging to decide what features should be implemented. Since the users had very different demands and the time span for developing the system was short, the requirements had to be segregated. A Scrum development framework was used to develop the whole system and its approach of dividing development into smaller units was a great help. Long dead ends were avoided because of having a fresh look at the previous progress and reconsidering priorities after each sprint.

Huerate system gradually evolved into a larger information system. It is developed in ASP.NET MVC web framework and runs in the Windows Azure cloud hosting. SQL Azure relational database and a NoSQL Azure Tables storage are used to store data. Its feedback form is optimized for smartphones and tablets web browsers, and administration provides complex customization capabilities for the restaurant owners.

The next steps should concentrate on finding out whether restaurant market in Czech Republic is ready to accept this kind of solution, because the level of restaurants involvement was unsatisfactory. The possible reason for such cold reception can be the way of presenting it as a student project. Another option worth exploring would be launching Huerate system in the western countries with bigger markets. The user interface is multilingual so the transition would be easy. Chapter 6 lists number of usability issues which were discovered during user testing. Those issues should also be fixed in order to provide as good user experience as possible.

However, restaurant owners who actively used Huerate system were very pleased with the results it provided. Few of them were able to fix a serious flaws in their restaurants after using Huerate for only two months. Therefore, the goal of the system was met and the thesis can be considered as successful

Bibliography

1. SCHMITT, B. *Customer experience management. A revolutionary approach to connecting with your customers*. New York: Wiley, 2003. ISBN 0-471-23774-4.
2. *SurveyMonkey: Free online survey software & questionnaire tool* [online]. [cit. 2013-January-06]. Dostupné z: http://www.surveymonkey.com/
3. TechCrunch. *SurveyMonkey Acquires MarketTools' Zoomerang, ZoomPanel, and TrueSample via TPG Capital* [online]. 14. December. 2011 [cit. 2013-January-05]. Dostupné z: http://techcrunch.com/2011/12/14/surveymonkey-tpg-markettools/
4. Freemium. *What is Freemium?* [online]. [cit. 2013-January-05]. Dostupné z: http://www.freemium.org/what-is-freemium-2/
5. SurveyMonkey. *Plans and Pricing* [online]. [cit. 2013-January-08]. Dostupné z: http://www.surveymonkey.com/pricing/upgrade/details/
6. SurveyGizmo - Affordable Enterprise Survey Software. *Online Survey Software* [online]. [cit. 2013-January-07]. Dostupné z: http://www.surveygizmo.com/
7. SurveyGizmo. *Online Survey Software Plans & Pricing* [online]. [cit. 2013-January-07]. Dostupné z: http://www.surveygizmo.com/plans-pricing/
8. Google Drive Help. *Forms* [online]. [cit. 2013-January-06]. Dostupné z: http://support.google.com/drive/bin/topic.py?hl=en&topic=1360904
9. doporucim.cz - Měření zákaznické spokojenosti a zkušenosti. *Ceník* [online]. [cit. 2013-January-02]. Dostupné z: https://www.doporucim.cz/cs/home/pricelist
10. Medallia. *Customers* [online]. [cit. 2013-January-06]. Dostupné z: http://www.medallia.com/customers
11. OSTERWEIL, W. The New Inquiry. *The Secret Shopper* [online]. 2012 [cit. 2013-03-14]. Dostupné z: http://thenewinquiry.com/essays/the-secret-shopper/
12. SCHWABER, K. a J. SUTHERLAND. The Scrum Guide. In: *The Definitive Guide to Scrum: The Rules of the Game* [online]. 2011 [cit. 2012-December-15]. Dostupné z: http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf
13. DONAGHY, B. P. Wired. *Where is Cloud Computing Headed in 2013?* [online]. 2012 [cit. 2013-05-15]. Dostupné z: http://www.wired.com/insights/2012/11/where-is-cloud-computing-headed-in-2013/

14. MELL, P. a T. GRANCE. *The NIST Definition of Cloud*. 2011. National Institute of Standards and Technology [cit. 2013-05-03]. Dostupné z: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
15. MICROSOFT. Windows Azure: Microsoft's Cloud Platform. *Introducing Windows Azure* [online]. [cit. 2013-02-03]. Dostupné z: http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/
16. MICROSOFT. Windows Azure: Microsoft's Cloud Platform. *Data Management* [online]. [cit. 2013-05-10]. Dostupné z: http://www.windowsazure.com/en-us/home/features/data-management/
17. MICROSOFT. Windows Azure: Microsoft's Cloud Platform. *Data Management and Business Analytics* [online]. [cit. 2013-03-12]. Dostupné z: http://www.windowsazure.com/en-us/develop/net/fundamentals/cloud-storage/
18. MICROSOFT. Windows Azure: Microsoft's Cloud Platform. *Windows Azure Pricing Calculator* [online]. [cit. 2013-05-14]. Dostupné z: http://www.windowsazure.com/en-us/pricing/calculator/?scenario=data-management
19. SENDGRID. SendGrid. *Product Pricing* [online]. [cit. 2013-02-21]. Dostupné z: http://sendgrid.com/static/pricing
20. MICROSOFT. Design and Implementation Guidelines for Web Clients. *Multithreading and Asynchronous Programming in Web Applications* [online]. [cit. 2013-05-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff647332.aspx
21. MICROSOFT. Microsoft Developer Network. *Partitioned Tables and Indexes* [online]. [cit. 2013-05-10]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms190787.aspx
22. MICROSOFT. Windows Azure: Microsoft's Cloud Platform. *Caching in Windows Azure* [online]. [cit. 2013-05-10]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsazure/gg278356.aspx
23. ULLRICH, J. ISC Diary. *Hashing Passwords* [online]. 2011 [cit. 2013-01-22]. Dostupné z: http://www.dshield.org/diary/Hashing+Passwords/11110
24. APACHE. Apache log4net. *Home* [online]. [cit. 2013-03-05]. Dostupné z: http://logging.apache.org/log4net/
25. MICROSOFT. The Official Microsoft ASP.NET Site. *MVC* [online]. [cit. 2013-05-18]. Dostupné z: http://www.asp.net/mvc
26. MICROSOFT. Microsoft Developer Network. *Model-View-Controller* [online]. [cit. 2013-03-09]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff649643.aspx

27. MICROSOFT. Microsoft Developer Network. *Walkthrough: Organizing an ASP.NET MVC Application using Areas* [online]. [cit. 2013-05-01]. Dostupné z: http://msdn.microsoft.com/en-us/library/ee671793(v=vs.100).aspx
28. SANDERSON, S. Knockout. *Home* [online]. [cit. 2013-05-11]. Dostupné z: http://knockoutjs.com/
29. CATLIN, H. N. WEIZENBAUM a C. EPPSTEIN. Sass - Syntactically Awesome Stylesheets. *Documentation* [online]. [cit. 2013-05-01]. Dostupné z: http://sass-lang.com/docs.html
30. MINDSCAPE. Web Workbench. *Sass, Less and CoffeeScript for Visual Studio* [online]. [cit. 2013-04-05]. Dostupné z: http://www.mindscapehq.com/products/web-workbench
31. BOGARD, J. AutoMapper - The object-object mapper. *Home* [online]. [cit. 2013-05-11]. Dostupné z: http://automapper.org/
32. MICROSOFT. Data Developer Center. *Get Started with Entity Framework (EF)* [online]. [cit. 2013-05-11]. Dostupné z: http://msdn.microsoft.com/en-us/data/ee712907
33. MICROSOFT. Data Developer Center. *Code First Migrations* [online]. [cit. 2013-05-11]. Dostupné z: http://msdn.microsoft.com/en-us/data/jj591621
34. KOHARI, N. Ninject. *Ninject - wiki* [online]. [cit. 2013-05-11]. Dostupné z: http://www.ninject.org/wiki.html
35. SCHWARZ, N. LUNGU a O. NIERSTRASZ. Seuss: Decoupling responsibilities from static methods for fine-grained configurability. Journal of Object Technology, 2012, č. Volume 11, no. 1 [cit. 2013-04-19]. Dostupné z: http://www.jot.fm/contents/issue_2012_04/article3.html
36. MICROSOFT. BizSpark. *Offers and Promotions for Microsoft BizSpark members* [online]. [cit. 2013-01-30]. Dostupné z: http://www.microsoft.com/bizspark/Offers/Default.aspx
37. ROGERS, E. *Diffusion of Innovations*. Glencoe: Free Press, 1962. ISBN 0-612-62843-4.
38. GROUP, N. N. Nielsen Norman Group. *User Testing* [online]. [cit. 2013-05-01]. Dostupné z: http://www.nngroup.com/topic/user-testing/
39. COHN, M. *Agile Estimating and Planning*. Prentice Hall, 2005. ISBN 978-0131479418.
40. MICROSOFT. Windows Azure: Microsoft's Cloud Platform. *Pricing Details - Cloud Services* [online]. [cit. 2013-05-15]. Dostupné z: http://www.windowsazure.com/en-us/pricing/details/cloud-services/

Appendix A Survey Results

Full results of the survey made among 315 Czech restaurants are shown in this appendix. Survey questions are displayed in Czech, because that was the language in which the survey was presented to the restaurants. Translating questions to English could have slightly changed meaning of questions what is undesirable.

A.1 Current ways of getting feedback

A.1.1 Jak teď získáváte zpětnou vazbu od Vaších hostů?

Obsluha se baví s hosty a názory pak předává majiteli	**259**	24%
Majitel se osobně baví s hosty	**228**	21%
Kniha přání a stížnosti v restauraci	**42**	4%
Možnost vyjádření názoru na stránkách restaurace	**131**	12%
Hosté píšou na Facebook	**164**	15%
Hosté posílají e-maily	**199**	19%
Tištěné dotazníky v restauraci	**41**	13%
Odkaz na elektronický dotazník v restauraci	**4**	1%
Jinak	**36**	3%

A.1.2 Kolik názorů Vaších hostů se k Vám přibližně dostane za měsíc všemi kanály (email, rozhovor,…)?

0-5 názorů měsíčně	**83**	27%
6-20 názorů měsíčně	**163**	52%
21-50 názorů měsíčně	**45**	14%
víc než 50 názorů měsíčně	**17**	5%
Jinak	**4**	1%

A.1.3 Jste spokojení s počtem a obsahem zpětné vazby od hostů?

Ne, rádi bychom dostávali víc názorů	**26**	43%
Ne, sice dostáváme dostatečné množství názoru, ale nedovíme se co potřebujeme	**2**	3%

Ne, chtěli bychom dostávat více názorů a s jiným obsahem	**7**	12%
Ano, s kvalitou a počtem zpětné vazby od hostů jsme spokojení	**22**	37%
Nevím	**3**	5%

A.1.4 Jak moc se ve směřování restaurace řídíte názory hostů?

1	**4**	1%
2	**22**	7%
3	**113**	36%
4	**95**	30%
5	**78**	25%

1. Minimálně - mám vlastní cestu a tou směřuji restauraci
5. Maximálně - snažím se co nejvíce vyjít vstříc hostům

A.1.5 Používáte mystery shopping?[9]

Ano	**15**	24%
Ne	**44**	71%
Nevím	**3**	5%

A.2 Huerate functions

A.2.1 Ocenili byste hodnocení jednotlivých jídel v rámci denního menu?

Ano, informace o kvalitě jednotlivých denních menu bych velmi ocenil(-a)	**170**	54%
Ano, tuto informaci bych ocenil (-a), ale není to moc důležité	**70**	22%
Ne, tato informace mě nezajímá	**10**	3%
Ne, neprovozujeme denní menu	**49**	16%
Jinak	**14**	4%

[9] This question was added later and only 62 respondents answered it

A.2.2 Jak moc Vás zajímají názory hostů v jednotlivých oblastech?

	Nezajímají (1)	Trochu zajímají (2)	Průměrně zajímají (3)	Dost zajímají (4)	Velmi zajímají (5)	Průměrná známka
Kvalita jídla obecně	2	2	6	39	264	4,8
Spokojenost s velikosti porce	3	9	50	74	177	4,3
Výběr jídel a pití	2	4	56	112	139	4,2
Hudba v restauraci	28	46	121	63	53	3,2
Obsluha obecně	0	1	6	40	266	4,8
Rychlost obsluhy	2	2	12	68	229	4,7
Ochota obsluhy	0	0	3	31	279	4,9
Prostředí restaurace	1	3	35	106	168	4,4
Hlasitost hudby	17	30	102	89	72	3,5
Pořádek na toaletách	4	1	20	60	228	4,6
Zakouřenost prostředí	43	4	36	65	165	4,0

A.2.3 Jakým způsobem byste chtěli sledovat odpovědi hostů?

Pravidelné emaily s přehledem odpovědí	**133**	37%
Možnost kdykoliv sledovat odpovědi na webu (v zaheslované sekci)	**194**	54%
Jinak	**33**	9%

A.2.4 Kolik času a úsilí byste byli ochotní věnovat přípravě a údržbě dotazníku?

Více než dvě hodiny týdně	**25**	8%
1-2 hodiny týdně	**78**	25%
Méně než hodinu týdně	**73**	23%
Jenom nezbytné minimum - chtěl(a) bych mít systém připraven k použití od začátku	**124**	40%
Jinak	**11**	4%

A.2.5 Chodí do Vaší restaurace i cizojazyční hosté?

Ano, často	**140**	46%
Ano, ale sporadicky	**154**	51%
Vůbec	**10**	3%

A.2.6 Byla by pro Vás zajímavá možnost nabídnout hostům malý dárek (nápoj, zákusek, …) za vyplnění dotazníku?

Ano, určitě bych to využíval (-a)	**49**	16%
Ano	**95**	31%
Ne	**91**	29%
Nevím	**74**	24%

A.2.7 Bylo by pro Vás užitečné získat e-mail hostů?

Ano	**220**	72%
Ne	**49**	16%
Nevím	**38**	12%

A.3 Information About Restaurant

A.3.1 Jakou má Vaše restaurace kapacitu?

Méně než 30 míst	**17**	5%
30 - 60 míst	**111**	35%
60 - 100 míst	**112**	36%
100 - 150 míst	**47**	15%
více než 150 míst	**26**	8%

A.3.2 Jaká je průměrná cena večeře ve Vaší restauraci?

do 100kč	**44**	14%
100 - 150kč	**128**	42%
150 - 250kč	**109**	36%
více než 250kč	**25**	8%

A.3.3 Je ve Vaší restauraci dostupná WiFi zdarma?

Ano	**218**	94%
Ne	**15**	6%

Appendix B Database Schema

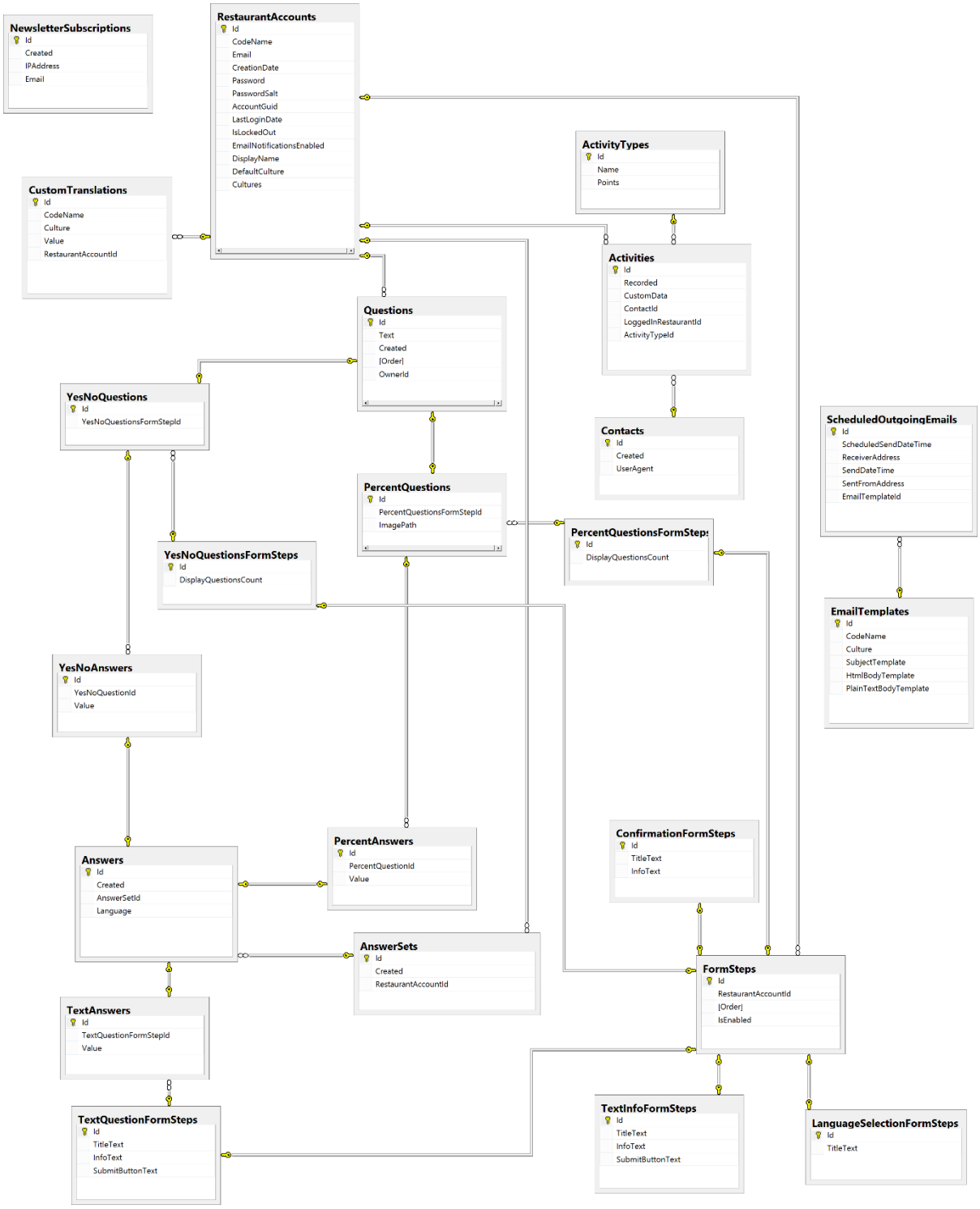


Figure 14: Huerate database schema

Appendix C Marketing E-mail

Dobrý den,

jmenuji se Jakub Kadlubiec a jsem studentem pátého ročníku Fakulty Informačních Technologií na VUT v Brně.

Rád bych Vás tímto požádal o pomoc s mou diplomovou prací.

Jako diplomovou práci tvořím službu pro restaurace pro sběr názorů a zpětné vazby hostů pomocí mobilních telefonů. Cílem je, aby vedení restaurace zjistilo, co hostům vadí a co se jim líbí. Na základě těchto informací je pak možné zlepšit služby v těch nejdůležitějších oblastech a tím nalákat více návštěvníků.

Obracím se na Vás se dvěma prosbami:

1. Velmi mě zajímá Váš názor jakožto potencionálního uživatele služby, a proto bych Vás rád poprosil o vyplnění krátkého dotazníku. Na základě výsledků budu dále směřovat svou práci. Vyplnění dotazníku zabere maximálně 5 minut. Najdete ho na adrese: https://docs.google.com/forms/d/1RlOgza42drfOLIrZ6XOSBqRcr78XScnRDd2AjtQohpI/viewform
2. Rád bych Vám nabídnul spolupráci ve formě zkušebního provozu ve Vaší restauraci zdarma. Další informace o tom, jak to vlastně funguje, a jak to můžete vyzkoušet, najdete níže v tomto mailu.

V systému Huerate (název mé diplomové práce) hosté odesílají zpětnou vazbu majiteli restaurace pomocí svých mobilních telefonů. V restauraci se umístí na stoly (například do jídelního lístku nebo jako samostatné letáky) www adresu. Host si tuto adresu otevře ve svém telefonu, zobrazí se mu dotazník a vyplní informace, které chce sdělit majiteli. Jak dotazník vypadá pro hosta se můžete podívat na adrese http://huerate.cz/vzor (nejlépe to vypadá na mobilním telefonu nebo tabletu).

Pokud máte jakékoliv dotazy ohledně fungování služby, tak mě neváhejte kontaktovat na jakub@huerate.cz.

Pokud máte zájem o zprovoznění ve Vaší restauraci, tak se můžete zdarma zaregistrovat na stránce http://huerate.cz/signup a pak postupovat podle návodu na http://huerate.cz/help. Všechno by mělo fungovat, ale pokud by byl přece jenom nějaký problém nebo by bylo potřeba asistence, tak mi neváhejte napsat na jakub@huerate.cz nebo zavolat na 739 661 033.

Předem mockrát děkuji za spolupráci,

Jakub Kadlubiec
jakub@huerate.cz

Appendix D Flyers at the Restaurants

Figure 15: Flyer with the Huerate link at the Country Saloon restaurant

Figure 16: Flyer with the Huerate link at the restaurant U Hrabětů